ioProgrammo

TUTTE LE NOVITÀ SU LINQ
IL LINGUAGGIO UNIVERSALE CREATO DA MICROSOFT PER L'INTERROGAZIONE DI QUALUNQUE TIPO DI DATO

Rivista + Le Grandi Guide di ioProgrammo n° 4 a € 14,90 in più

VERSIONE PLUS ✔ RIVISTA+LIBRO+CD €9,90 — VERSIONE STANDARD ☐ RIVISTA+CD €6,90

PER ESPERTI E PRINCIPIANTI

Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL Periodicità mensile • GENNAIO 2006 • ANNO X, N.1 (98)

PHP BLOG
Crea un tuo engine per la costruzione di un sistema di gestione dei contenuti

Skinnable: l'interfaccia varia modificando normali file HTML

Estendibile: con moduli semplici da programmare e installare

Velocissimo: con le stored procedure di MySQL5

PDF FACCIAMOLI CON VISUAL BASIC
Produci report leggerissimi direttamente dal tuo codice e senza bisogno di altro software aggiuntivo

JAVA E XML VELOCI COME NON MAI!
Impara come leggere e scrivere i dati in un formato standard e facilmente portabile da un'applicazione all'altra, con prestazioni da competizione

■ C#
FATTI L'ADD-IN PER VISUAL STUDIO
Ecco come estendere l'ambiente con le funzionalità che non ci sono nella versione standard

■ VISUAL BASIC.NET
DA DATABASE AD EXCEL SENZA PROBLEMI
Usa XSL per trasportare i dati da un qualunque formato al foglio di calcolo usato da Microsoft

.NET
PROGRAMMA IL TUO CATTURA SCHERMO
Utilizza le API di sistema per fare le cose che normalmente .NET non fa

IMMAGINI BITMAP SOTTO CONTROLLO
Tutte le tecniche per caricarle, trasformarle e gestirle al meglio

JAVA
AUTENTICHIAMO GLI UTENTI
Ecco come concedere solo a chi vuoi tu l'accesso alla tua applicazione o al tuo portale

GESTIAMO L'ANAGRAFICA CLIENTI
Usiamo iBatis per "mappare" le tabelle SQL con classi ed oggetti e rendiamo tutto più semplice

SPECIALE
ioProgrammo BY EXAMPLE
30 PROBLEMI RISOLTI
CON GLI ESEMPI DI CODICE RAPIDO DA COPIARE E INCOLLARE

PER TUTTI I LINGUAGGI: VISUAL BASIC, VISUAL BASIC.NET, C#, PHP, JAVA, JAVASCRIPT, ASP.NET, HTML

SECURITY: dal classico problema della fattorizzazione al suo utilizzo all'interno dell'algoritmo RSA

EDIZIONI MASTER
www.edmaster.it

9 771128 594641 60098

Anno X - N.ro 1 (98) - Gennaio 2006 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Redazione: Fabio Farnesi
Collaboratori: M. Autiero, M. Bigatti, L. Buono, F. Grimaldi, M. Locuratolo, A. Pelleriti, R. Sidoti, F. Smelzo.

Segreteria di Redazione: Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Aurelio Monaco

Realizzazione Multimediale: SET S.r.l.
Coordinamento Tecnico: Piero Mannelli
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via C. Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Tel. 02 831213 - Fax 02 83121330
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti

ABBONAMENTO E ARRETRATI

ITALIA: Abbonamento Annuale: ioProgrammo (11 numeri) €59,90 sconto 20% sul prezzo di copertina di €75,90 · ioProgrammo con Libro (11 numeri) €75,90 sconto 30% sul prezzo di copertina di €108,90
Offerte valide fino al 31/03/06
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via C. Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato.
La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via C. Correnti, 1 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
tel. 02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Arti Grafiche Boccia S.p.a. Via Tiberio Felice, 7 Salerno
Stampa CD-Rom: Neotek S.r.l. - C.da Imperatore - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Dicembre 2005

Calcio & Scommesse, Colombo, Computer Bild Italia, Computer Games Gold, Digital Japan Magazine, Digital Music, DVD Magazine, Filmteca in DVD, Giochi e Programmi per il tuo telefonino, Go!OnLine Internet Magazine, Guide di Win Magazine, Guide Strategiche di Win Magazine giochi, Home Entertainment, Horror mania, I Corsi di Win Magazine, I Fantastici CD-Rom, I film di idea web, I Filmissimi in DVD, I Libri di Quale Computer, I Mitici all'Italiana, Idea Web, InDVD, IoProgrammo, Japan Cartoon, La mia Barca, La mia Videoteca, Le Grandi Guide di ioProgrammo, Linux Magazine, Miami Vice in DVD, MPC, Nightmare, Office Magazine, Popeye, PC Junior, PC VideoGuide, Quale Computer, Softline Software World, Supercar in dvd, Thriller Mania, Win Junior, Win Magazine Giochi, Win Magazine, Le Collection.

EDIZIONI MASTER
A.N.E.S.
Certificato n.5330 del 2/12/2004
ITportal L'Universo Tecnologico www.itportal.it

Imparare con gli esempi

In questo numero di ioProgrammo, trovate una grande novità, l'inserto "ioProgrammo by Example". Si tratta di una grande novità per chi ci legge abitualmente. L'idea è molto semplice ed è una risposta diretta alle centinaia di email che ogni giorno giungono alla nostra redazione. Si, sono importanti le grandi tecniche, quelle che ioProgrammo affronta e continuerà ad affrontare come fa da dieci anni a questa parte, con il vanto di avere contribuito a formare una diversa generazione di programmatori. Ma queste tecniche non sono niente se non sono accompagnate da una conoscenza diretta degli strumenti, della sintassi, delle problematiche più semplici che spesso e volentieri al di la della pura tecnica ci fanno perdere tempo nella programmazione giornaliera.
C'è una seconda ragione, forse più grande che ci spinge a dedicare una parte importante della rivista a questo aspetto ed è che "ioProgrammo è certa che l'Italia ha bisogno di programmatori". La nostra generazione, quella nata nelle università degli anni 80/90 è quella che attualmente sostiene le software house italiane. Ma dove sono i prossimi programmatori? quelli che attualmente frequentano le università sono persi in fumose discussioni accademiche che spesso messe a confronto con le necessità delle aziende si risolvono in lunghissimi tempi di formazione, prima di poter giungere a divenire "Programmatori produttivi". Coloro i quali, freschi di laurea, affrontano progetti concreti spesso non hanno quella abitudine a scrivere codice che contraddistingue chi lavora da tempo. E allora ioProgrammo ancora una volta si vuole assumere, come ha fatto già in precedenza, la responsabilità di contribuire alla formazione di una nuova generazione di programmatori, che padroneggi le tecniche di grande respiro ma che sia anche in grado di risolvere problemi concreti. Da qui l'invito ancora una volta a formare una community di sviluppatori italiani dove ciascuno mette a disposizione degli altri le proprie conoscenze. Inviateci i vostri quesiti, e per i più esperti inviateci le vostre soluzioni, saremo ben felici da fare da anello di raccordo per aiutare il settore della programmazione in Italia a crescere. Per tutti l'email è ioprogrammo@edmaster.it

Fabio Farnesi ffarnesi@edmaster.it

CD WEB
nome_file.zip
cdrom.ioprogrammo.it

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.

PHP BLOGS

Crea un tuo engine per la costruzione di un sistema di gestione di contenuti

- ✔ **Skinnable: l'interfaccia varia modificando normali file HTML**
- ✔ **Estendibile: con moduli semplici da programmare e installare**
- ✔ **Velocissimo: con le stored procedure di MySQL5**

pag. 14

PDF? FACCIAMOLI CON VISUAL BASIC

Crea report leggerissimi direttamente dalla tua applicazione e senza bisogno di software aggiuntivo pag. 86

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

Prodotti del mese

MySQL 5.0

Finalmente la nuova versione!

La più innovativa di sempre

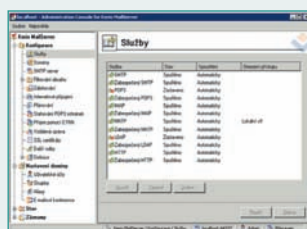
Gli sviluppatori da sempre hanno fatto una colpa a MySQL per l'assenza di strumenti quali stored procedure e trigger. Pur essendo il database più usato in ambiente web, pur essendo estremamente semplice e veloce, riferendosi a MySQL si è sempre detto che l'assenza di questi strumenti non lo configurava come un database professionale. Da questa versione in poi non si potrà più dire niente del genere. MySQL 5.0 rappresenta un salto di continuità rispetto al passato. Sono stati aggiunte proprio le funzioni relative a Stored Procedure e ai Trigger, che vanno a completare la nutrita schiera di funzionalità che già caratterizzava il prodotto. Infine si deve citare anche l'introduzione del concetto di view. Se non avete già pensato a una migrazione dalle vecchie versioni forse è il caso di farlo!

[pag.107]

Kerio Mailserver 5.7.1

Completo e affidabile

Davvero ben fatto questo Mail Server prodotto da Kerio. È proprio la completezza il punto di forza di questo prodotto. Prima di tutto si deve considerare l'altro numero di protocolli supportati. SMTP, POP3, Secure POP3, IMAP, Secure IMAP, Webmail /WAPmail, SecureWebmail /WAPmail. Una rapida occhiata nel comodo pannello di amministrazione ci informa anche che il prodotto è fortemente integrato con Spam Assasin e con il motore Antivirus di Mcaffee. Le altre caratteristiche interessanti sono costituite da un sistema di backup che consente di effettuare una copia del sistema di posta ad intervalli predefiniti, e dallo scheduler che consente di gestire la coda degli invii in modo ottimale.
Una nota particolare merita il plugin per la stampa delle mail su un fax remoto. Infine la webmail integrata risulta molto comoda;

[pag.106]

NHibernate 1.0.10

La persistenza dei dati

Versione .NET

Hibernate è uno dei tool di persistenza più noti al mondo. Consente di mappare oggetti a dati tipicamente strutturali come quelli esposti da SQL, e infine di mantenerli persistenti. NHIbernate è il porting di questo software per la piattaforma .NET.
Un'applicazione che certamente, nonostante, una certa complessità, risolve alcuni dei problemi che maggiormente affliggono i programmatori in fase di stesura di applicazioni che si connettano a DB. L'efficacia di un prodotto come NHibernate si misura dalla sua capacità di innalzare la produttività di chi lo utilizza. D'altra parte é noto che la programmazione ad oggetti rappresenta la soluzione ad una serie complessa di problemi. Poterla applicare ai database rappresenta sicuramente un enorme vantaggio.

[pag.108]

Acqua Data Studio 4.5.2

Il SQL Manager è servito

Acqua Data Studio è uno strumento di decisamente evoluto. Scritto in Java si configura come un SQL Manager completo e affidabile. Supporta un gran numero di DBServer, da Oracle a MsSQL a MySQL.
Consente la gestione totalmente grafica delle tabelle, dei database, delle interrogazioni SQL. Nel nostro caso lo abbiamo usato in alternativa a PHPMyAdmin, trovandolo utile come sistema da utilizzare in modo standalone, ancora leggermente inferiore per quanto riguarda il numero di funzionalità esposte. D'altra parte PHPMyAdmin è un fuoriclasse nel suo genere ed è tarato su MySQL, mentre AcquaData Studio è del tutto generico. In ogni caso si tratta di un sistema che vi può trarre fuori dai guai in più di una situazione. Da provare!

[pag.105]

News

AL VIA SIMPLICITY IL RAD PER JAVA

Data Representations Inc si lancia con coraggio in un mercato dominato da nomi altisonanti quali Borland Jbuilder o NetBeans. Lo fa con un insieme di software commerciali che si propongono di diventare un punto di riferimento negli ambienti di sviluppo rapido destinati al sistema di programmazione di Sun. Simplicity è disponibili in diverse versioni. Fra le più interessanti citiamo "Simplicity for Mobile" destinato allo sviluppo di applicazioni per cellulari e palmari. Accanto a questo si pongono Simplicity for Enterprise dedicato come è ovvio allo sviluppo di applicazioni distribuite e infine Simplicity per Desktop che rappresenta l'entry point per chi sviluppa applicazioni destinate a essere usate in ambienti single user.
Il nodo centrale di Simplicity è comunque costituito dalla sua capacità di essere WYSIWYG ovvero di fornire all'utente un ambiente di disegno delle interfacce collegato con il classico sviluppo ad eventi che ha fatto la fortuna di ogni ambiente RAD che si rispetti.

LA RISCOSSA DI OS2

Il sistema operativo di IBM è stato per anni sogno e speranza nascosta di ogni programmatore. E per anni ha fatto il bello e il cattivo tempo all'interno di ambienti delicati come banche, magazzini, uffici, grande industrie, fino a quando IBM ne ha cessato lo sviluppo con immenso dispiacere degli affezionati, che hanno visto così crollare un vero mostro di tecnologia quale era OS2. Tuttavia il popolo dei programmatori non poteva far morire così una pietra miliare dei sistemi operativi al mondo, ed è così nato osFree, ovvero il tentativo di sviluppare un sistema operativo compatibile con OS/2 Merlin. Attualmente il progetto è del tutto in fase embrionale e funzionano pochi tool a riga di comando ma sta già risollevando gli entusiasmi di tantissimi appassionati. Chi volesse partecipare allo sviluppo o semplicemente avere notizie su questa idea entusiasmante può fare riferimento a:

www.osfree.org/index.php?gobaby=status

MICROSOFT ADOTTERÀ FORMATI APERTI

Si chiamerà Office Open Xml il nuovo formato "aperto" certificato da Ecma e che caratterizzerà le nuove versioni di Microsoft Office. Si tratta di un passo incredibilmente rivoluzionario per quanto riguarda Microsoft, dettato dalle pressioni della commissione Europea, e da quella effettuata dai governi internazionali che non vedono di buon occhio l'utilizzo di formati proprietari. In realtà Microsoft nega che questo passaggio sia dovuto a una crescente necessità di trasparenza nell'uso del software da parte delle amministrazioni pubbliche, piuttosto lo giustifica mettendo in rilievo il crescente impegno che la casa di Redmond ha adottato nello spingere i formati aperti basati su XML. Infatti il nuovo formato sarà ovviamente basato su XML e

È ACCORDO FRA SUN E POSTGRESQL

La prossima release di Solaris sarà profondamente integrata con il database PostgreSQL. L'annuncio è di quelli che scottano, non tanto a causa dell'annunciata integrazione, quanto perché con questa mossa Sun mette un piede all'interno di un settore strategico del mercato, quello dei database. La collaborazione con PostgreSQL di fatto non si limiterà alla sola integrazione del DB in Solaris, piuttosto vedrà la partecipazione attiva degli sviluppatori di Sun all'interno del team dei progettisti di PostgreSQL. Se si considera che già ora Postgres si può considerare come un database altamente professionale dotato di strumenti fuori del comune dello stesso livello di Microsoft SQL Server o Oracle si intuisce come le potenzialità derivanti dall'immissione di nuove e capaci forze all'interno del team di sviluppo, rappresenta un punto di svolta senza precedenti nel settore strategico dei DB. Staremo a vedere quello che ci proporranno non solo le nuove release di Solaris ma anche e soprattutto i prossimi rilasci di PostgreSQL.

si propone di mantenere una piena retrocompatibilità con i formati precedenti. Prima di Microsoft già OpenOffice aveva proposto uno standard aperto per i propri documenti: l'OpenDocument. Un file .odt di OpenOffice è un jar rinominato contenente al suo interno una serie di documenti XML che ne descrivono il contenuto.
Vedremo come sarà elaborato il nuovo formato di Microsoft. Resta il fatto che si tratta di una svolta storica, che molto probabilmente influenzerà l'intero mercato delle applicazioni per ufficio a partire dalla prima prossima futura release del nuovo Office.

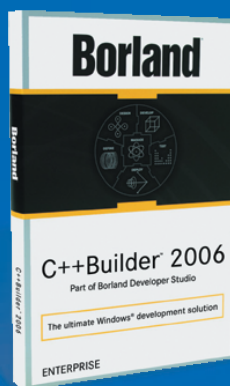
ARRIVA DEVPARTNER PER VISUAL STUDIO 2005

In parallelo con il lancio di Microsoft Visual Studio, Compuware ha annunciato il rilascio di nuove versioni dei suoi prodotti DevPartner, Compuware DevPartner Studio 8.0 e Compuware DevPartner Fault Simulator 1.5. Ovviamente il focus principale di queste nuove versioni è l'integrazione com Visual Studio 2005. Le soluzioni di Compuware si pongono come uno strumento prezioso per il debugging delle applicazioni durante tutto il loro ciclo di vita. In particolare DevPartner Studio 8.0 e DevPartner Fault Simulator si integrano alla perfezione con Visual Studio 2005 Team System, consentendo all'intero team di sviluppo di partecipare in modo attivo anche alla fase di testing e debugging del software.
Viceversa DevPartner Studio 8.0 Professional Edition è pensato per lavorare alla perfezione con Microsoft Visual Studio 2005 nella sua versione pensata per il programmatore singolo. Una delle innovazioni più importanti sarà il Fault Simulator SE che consente di effettuare una simulazione degli errori facilitando enormemente il processo di testing delle applicazioni.

RILASCIATO MONO 1.1.0

In coincidenza con la presentazione di Microsoft Visual Studio 2005, con una puntualità che potrebbe far pensare male è stata rilasciata la versione 1.1.0 dell'implementazione OpenSource di .NET ovvero MONO.
La nuova versione aderisce alle specifiche di C# 2.0 eccetto per alcune particolarità riferite alla non completa implementazione delle Windows Forms, motivo per cui la versione in questione prende numerazione 1.1.0 e non 2.0. Per quanto discusso, Mono sta riscuotendo un notevole successo in ambiente Linux e rappresenta una possibilità interessante per coloro che vogliono sviluppare applicazioni multipiattaforma sia partendo da un lato che dall'altro. Quello che sorprende è che questa possibilità non sia ancora stata sfruttata a fondo, di fatto le versioni del software scritte per Linux trovano una scarsa corrispondenza in Windows e viceversa.
Il degno avversario Java invece sfrutta a fondo la sua natura multipiattaforma. Quanto ancora dovremo aspettare perché il mondo della programmazione decida di non lavorare a compartimenti stagni?

PRESENTATI DELPHI E C++ BUILDER 2006

Una grande novità per gli affezionati dei tool di casa Borland. I nuovi IDE sono arrivati e promettono eccezionali migliorie nel campo delle produttività per le applicazioni Mission Critical. Sostanzialmente si tratta di un'evoluzione della già ottima versione 2005 che evolve nel senso di una maggiore stabilità nella programmazione delle GUI dei Database e nello sviluppo Web Oriented che assume un ruolo centrale ed è supportato dal concetto di WYSIWIG tanto caro ai rad di casa Borland. Sono stati aggiunti nuovi providers verso tutti i database attualmente disponibili, una maggiore integrazione con CVS, e alcuni tool interessanti di Bug Tracking, infine sono significative le migliorie apportate al sistema collaborativo che consente di sviluppare applicazioni in team. al punto di vista più strettamente tecnico, la VCL si presenta sempre come uno dei repository di componenti più completi nel campo dei framework, e tuttavia è possibile utilizzare i prodotti di Borland anche per accedere direttamente a .NET e sviluppare in questo modo con le tecnologie tanto care a Microsoft. Il resto è il solito IDE assolutamente completo proposto da casa Borland, Wisywig, drag & drop, interprete del concetto di RAD nella maniera più estrema possibile.

PHP Blogs

In questo articolo vedremo come costruire un motore per CMS che vi consenta di sviluppare siti skinnable modulari e estendibili. Insieme vedremo qualcosa del nuovo MySQL 5

La nostra idea è quella di costruire un CMS engine in PHP. Per CMS Engine intendiamo uno scheletro che metta in condizione l'utente di avere a disposizione un framework personalizzato per la costruzione di un qualunque CMS. La differenza con migliaia di CMS in circolazione è evidente. Il nostro articolo non sarà incentrato sulla scrittura di funzioni per l'inserimento e la raccolta dei dati. Un qualunque buon manuale su PHP sarebbe in grado di spiegarvi come inserire e recuperare dei dati da MySQL! Noi costruiremo un framework tale che il programmatore dovrà solo occuparsi di:

a) affidare a un grafico il disegno dell'interfaccia grafica;
b) programmare "moduli" secondo una certa logica.

Il risultato finale sarà che l'interfaccia grafica e la logica di programmazione saranno completamente separate. Al grafico sarà sufficiente inserire un tag **{module}** all'interno di puro codice HTML per ottenere in output i dati gestiti da un particolare modulo. Un modulo potrebbe essere un calendario, piuttosto che un banner, piuttosto che restituire tutte le news dell'ultimo mese, oppure consentire di visualizzare immagini etc. Il grafico non saprà mai come sono sviluppati i moduli. A lui interesserà solo sapere che un modulo esiste e che può essere usato per raggiungere un certo scopo. Il programmatore si occuperà di definire i moduli con una logica che sarà dettata da un engine.
Sostanzialmente l'engine ha funzioni di raccordo. Da un lato carica i moduli, dall'altro prende i dati da un modulo e li trasforma in tag utilizzabili da un web designer che non abbia conoscenza di PHP o di un qualunque altro linguaggio che non sia HTML e poco altro.

REQUISITI

Conoscenze richieste
Basi di programmazione PHP

Software
PHP 5, MySQL 5

Impegno

Tempo di realizzazione

INSTALLAZIONE DI SMARTY

È sufficiente scaricare i file che costituiscono il progetto da http://smarty.php.net, scompattarli e aggiungere un path di inclusione nel file *php.ini*.

Ad esempio:

```
include_path = "include_path =
        ".:/usr/share/php:/usr/share/php
                    /Smarty-2.6.10/libs"
```

CARATTERISTICHE DEL BLOG

Il nostro Blog dovrà avere tutte le caratteristiche di un normale CMS più qualche funzionalità che definiamo di seguito:

1) Deve essere Skinnable, ovvero deve poter cambiare completamente interfaccia grafica semplicemente cambiando i file Html che ne definiscono l'aspetto, senza per questo dover intervenire sulla parte programmativa
2) Deve essere modulare, ovvero deve consentire di caricare nuove funzionalità semplicemente inserendo il file che le definisce all'interno di un'apposita directory. I "Moduli" che definiscono una funzionalità possono avere un output sotto forma di blocco che può essere inserito in un qualunque punto dello skin.

COME SARÀ COMPOSTO IL NOSTRO PROGETTO?

Divideremo il lavoro in tre parti fondamentali. Per quanto riguarda l'interfaccia grafica ci affideremo a Smarty. Ovvero il template engine di PHP. Svilupperemo poi un engine che "scansionerà" una directory alla ricerca di file con estensione *.blocks* e, se ne trova, li utilizzerà in modo da restituire a smarty un tag utilizzabile. Infine utilizzeremo una classe d'appoggio denominata *"dbase"* per eseguire compiti ripetitivi di recupero dati dal db.
Ora tutti sanno come funziona smarty. Ovvero all'interno di un file .PHP viene instanziato un oggetto di classe smarty. A questo oggetto vengono fatti degli "Assign" ad esempio:

```
smarty->assign('hello','hello world');
```

viene associato all'oggetto un template:

```
smarty->display(index.tpl):
```

se all'interno del file *index.tpl* verrà trovato un tag **{$hello}** esso viene espanso con il corrispondente *'Assign'* e viene restituito l'output ad esso associato,

nel nostro caso 'hello world'. Va da sé che il nostro scopo è generare una serie di assign per il template che stiamo usando. Il nostro problema è che non conosciamo a priori gli assign, che invece devono essere recuperati dai moduli che ogni sviluppatore programmerà per un preciso scopo. Se avete seguito attentamente i passaggi, avete già capito che ogni modulo deve avere una funzione che assegni il suo output a una variabile comprensibile dal template. Allo stesso modo, sapete che vogliamo sviluppare un framework; senza costringere ogni sviluppatore a interfacciarsi con smarty, questo sarà compito dell'engine.
Vediamo come funzionerà praticamente il tutto.

IL BOOTSTRAP DEL SISTEMA

Il file principale *index.php* conterrà poche ma fondamentali istruzioni:

```
<?php
//include il file delle classi smarty
include_once('Smarty.class.php');
//include il file del controller
include_once 'includes/controller.php';
// inizializza il controller
$mycontroller= new controller;
?>
```

Il primo file di cui andremo ad occuparci è dunque *controller.php*. Il nostro controller sarà costruito in modo da facilitarci tutti i prossimi lavori, il codice proposto è il seguente:

```
<?
class controller {
public $vista;
public $modello;
function controller() {
      $this->vista = new smarty;
      $this->load_all_blocks();
      $this->vista->display('index.tpl');}

private function RemoveExtension($strName)
   {  $ext = strrchr($strName, '.');
       $strName = substr($strName, 0, -strlen($ext));
       return $strName; }
   private function factory($type) {
      $classname = $type;
      $operation=$_GET['operazione'];
      $modules=$_GET['modules'];
      return new $classname($modules,$operation);}
   private function load_all_blocks()
  { if ($handle = @opendir('./modules')) {
         while (false !== ($file = readdir($handle))) {
            $type=$this->RemoveExtension($file);
                 if ( $file !== ".." & $file !== ".") {
                    include_once "./modules/$type.blocks";
                    $classname = $this->factory($type);
                    if (method_exists($classname,'operation')) {
                          $this->vista->assign($type.'operation',
                                      $classname->operation());}
                     if (method_exists($classname,"output")) {
                         $this->vista->assign($type,
                                         $classname->output());}
    }  } } }
}
```

Questo semplice file costituisce il nostro engine, il cuore del sistema. La classe controller espone una proprietà pubblica che si chiama "vista". Questa proprietà viene immediatamente riempita instanziando un oggetto di tipo smarty, all'interno del costruttore. Il nostro oggetto "vista" di classe smarty rappresenta il nostro contenitore per tutto ciò che concerne l'interfaccia grafica. Fisicamente viene associato al file index.tpl tramite l'istruzione:

```
$this->vista->display('index.tpl');
```

Questa istruzione fa sì che il compilatore PHP carichi in memoria il file *index.tpl*, lo parserizzi e se incontra al suo interno dei tag di tipo *{xxxx}* espanda il loro contenuto secondo certe regole che vedremo fra un momento. Ancora nel costruttore viene richiamato il metodo *load_all_blocks();*. Questo metodo per certi versi si può definire come un centro nevralgico dell'intero progetto. Diamogli uno sguardo da vicino:

```
private function load_all_blocks()
{ if ($handle = @opendir('./modules')) {
      while (false !== ($file = readdir($handle))) {
         $type=$this->RemoveExtension($file);
         if ( $file !== ".." & $file !== ".") {
            include_once "./modules/$type.blocks";
            $classname = $this->factory($type);
            if (method_exists($classname,'operation')) {
               $this->vista->assign($type.
                 'operation',$classname->operation());}
            if (method_exists($classname,"output")) {
               $this->vista->assign($type,$classname->
                                 output()); } }}}
}
```

Viene scansionata l'intera directory modules alla ricerca di file con estensione .blocks, se uno di questi file viene trovato, esso viene incluso tramite un include, e immediatamente dopo viene creata un'istanza della classe contenuta nel modulo tramite il metodo factory:

```
private function factory($type) {
            $classname = $type;
```

NOTA

IL PATTERN MVC

Un pattern è un modello di programmazione, ovvero uno schema ben studiato e definito seguendo il quale si ottengono risultati apprezzabili. Per quanto riguarda il ben noto pattern MVC si tratta dell'acronimo di *Model /View/Controller*. Ovvero un modello che suggerisce di dividere un progetto in tre parti fondamentali, una che si occupi di definire il "Modello" dell'applicazione, una che si occupi di definire la sua parte visibile all'utente finale, ovvero l'interfaccia grafica, e una che svolga funzioni di controllo, cioè gestisca i vari eventi che costituiscono il flusso del programma. Il pattern MVC è leggermente più complesso rispetto a quanto abbiamo fin qui detto, ma per i nostri scopi la definizione è più che corretta.

PHP

```
        $operation=$_GET['operazione'];
        $modules=$_GET['modules'];
     return new $classname($modules,$operation);
}
```

Notate che alla classe vengono passati nel costruttore le due variabili globali *modulo* e *operazione* che utilizzeremo in un secondo momento. Infine viene controllata l'esistenza di un metodo *'output'* nell'oggetto appena instanziato. Se questo metodo viene trovato, il suo risultato viene assegnato all'oggetto *$vista* che abbiamo instanziato prima, inizializzando il suo nome al nome del modulo e il suo valore al valore di ritorno del metodo output. L'altro metodo che viene controllato è il metodo "operation". Operation conterrà il nome dell'operazione che andremo a svolgere e passerà questo nome al template nella forma *modulooperazione,* questo ci consentirà di sviluppare dei template "condizionali" come avremo modo di vedere a breve. Pertanto, supposto che nella directory *modules* esista il modulo *dummy.blocks,* esso potrebbe contenere:

```
<?
class dummy {
        private $op;
        private $modules;
        function dummy($modules,$operation){
                $this->op=$operation;
                $this->mod=$modules;}
        public function output() {
                return "i'm a dummy module";} }
?>
```

A questo punto basterà inserire in *index.tpl* il tag *{$dummy}* per ottenere l'effetto desiderato, richiamando *http://localhost/* per ottenere in output la stringa

```
i'm a dummy module
```

Il nostro engine è pronto. *load_all_blocks* ha scansionato la directory *modules,* ha trovato il file *dummy.block,* lo ha incluso, ha creato un oggetto di classe dummy, ha controllato al suo interno il metodo output e ha passato l'output del modulo all'oggetto di tipo smarty, chiamando la variabile smarty con lo stesso nome del modulo! Se avrete l'accortezza di leggere con attenzione il codice e seguire il suo flusso scoprirete che è molto più semplice da comprendere se proverete voi stessi a riscrivere il codice all'interno di un vostro progetto di prova. Abbiamo ottenuto il primo passo, ovvero l'indipendenza dell'interfaccia dalla logica di programmazione, sarà sufficiente adesso sviluppare i moduli *.blocks,* che abbiano almeno il metodo output e inserire i corrispondenti tag nel file *.tpl* per poter creare qualunque tipo di interfaccia.

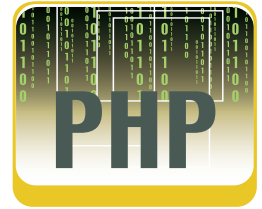
IL DATABASE

Il nostro prossimo passo è sviluppare un modulo che si chiama story. Il corrispondente tag nel file .tpl deve mostrare l'elenco degli ultimi post effettuati sul sito. Di conseguenza, il nostro modulo dovrà connettersi al db, effettuare una select, restituire in output l'elenco richiesto. È proprio su questo elenco che ci soffermeremo. Di fatto, mentre il semplice blocco "dummy', sviluppato in precedenza, restituiva una stringa, in questo caso vogliamo ottenere un array associativo che sfrutteremo in smarty con una tecnica che vi mostreremo da qui a breve. Prima di ogni cosa però pensiamo a come dovrà essere costruita una riga della tabella story nel nostro database. I campi che ci vengono in mente sono:

```
[id_stpria] – [id_autore] – [story]
```

È molto banale ma, per il nostro esempio, va più che bene. Se volete potrete continuare a espandere questo progetto per i vostri scopi in futuro come meglio desiderate. Poiché abbiamo già sviluppato l'engine, non ci resta che concentrarsi sul modulo story. Una prima implementazione molto grezza è la seguente:

```
<?
<?
include_once('includes/database.php');
class story {
  private $op;
  private $mod;
  private $data=null;
  function story($modules,$operation){
    $this->op=$operation;
    $this->mod=$modules;
    if ($modules == 'story') {
      $this->switchoperation();
    } else {
      $query='call selectstory();';
      $this->data=$this->doquery($query); } }
  private function switchoperation() {
    $query='call selectstory();';
    $this->data=$this->doquery($query);}
  private function doquery($query) {
    $db = new dbase;
    $data = $db->execquery($query);
    return $data;}
  public function operation() {
    return $this->op;}
  public function output() {
    return $this->data;}}
?>
```

L'engine instanzia l'oggetto 'story', tramite il metodo factory. Viene richiamato il costruttore di story che dopo alcuni controlli, chiama il metodo 'doquery' passandogli come parametro 'call selectstory()'. Questo metodo esegue una query sul database e

scrive il risultato nella variabile data. Infine il metodo output restituisce il contenuto di 'data', che viene usato dall'engine per costruire il tag relativo a story. Notate che il modulo in questione fa uso di una classe dbase contenuta in database.php. Questa classe fa parte del model e conterrà tutto il necessario per interfacciare il nostro software al database. Una prima semplice implementazione è la seguente:

```
<?php<?php
class dbase {
  private $server = "localhost";
  private $database = "ioPcms";
  private $username = "root";
  private $password = "xxxx";
  public $db=null;
  function dbase() {;
    $this->db = new mysqli($this->server, $this->
     username, $this->password, $this->database);}
  function execquery($query) {
    $result = $this->db->query($query) ;
    if ($result->num_rows>0) {
      while ($rs=$result->fetch_assoc()) {
        $temp[]=($rs);}}
    return $temp;}}
?>
```

In sostanza il metodo output del modulo story, che sarà anche quello che deve ritornare l'array associativo da usare nel template, utilizza il metodo execquery della classe dbase per ottenere l'elenco delle storie presenti nel database.

STORED PROCEDURE

I più attenti di voi si saranno già accorti che abbiamo passato al metodo *execquery* una query assolutamente assurda, ovvero

```
call selectstory();
```

È un comando del tutto sconosciuto agli utilizzatori di MySQL. Di fatto stiamo usando una delle nuove funzionalità di MySQL5, ovvero le *stored Procedure*. Una *stored Procedure*, è una sorta di subroutine embedded nel database e dotata di un'etichetta.
Ad esempio:

```
DELIMITER $$
DROP PROCEDURE IF EXISTS
                    `ioPcms`.`selectstory`$$
CREATE PROCEDURE `selectstory`()
BEGIN
  select * from story order by id_story DESC limit 0,30;
END$$
DELIMITER ;
```

queste sono le istruzioni per creare la stored procedure selectstory, così come l'abbiamo usata noi all'interno del nostro progetto. Si tratta di una story procedure immagazzinata direttamente all'interno del database e che restituisce l'elenco dei record della tabella story ordinati secondo un certo criterio. I vantaggi dell'utilizzo di una stored procedure sono notevoli. Prima di tutto ci consente di "atomizzare" ancora la nostra applicazione. Significa che possiamo limitarci a scrivere il codice, il grafico può fare il suo mestiere usando i template, un progettista SQL si può occupare di definire le stored procedure che noi utilizzeremo all'interno del nostro progetto. L'uso delle stored procedure velocizza moltissimo le operazioni, in quanto la stored procedure è contenuta all'interno del database stesso e in caso di query complesse o comandi multipli questo comporta una maggiore efficacia di una stored procedure rispetto a una query esterna. Infine l'uso delle stored procedure consente una maggiore sicurezza.

ARRAY ASSOCIATIVI

Abbiamo ottenuto il nostro risultato. Siamo in possesso di un tag *{$story}* da poter inserire nel nostro file *.tpl*. Se lo facciamo però ci accorgiamo di non ottenere la risposta corretta, bensì la nostra pagina stamperà semplicemente: *Array*. Questo accade perché abbiamo passato al tpl un dato complesso formato da un array associativo piuttosto che semplicemente una stringa. Abbiamo preferito questo metodo perché nella maggior parte dei casi il database ci restituirà un dato complesso, costituito ad esempio da

Titolo
Testo Breve
Testo Lungo
Autore
Data

Per il nostro esempio e per brevità abbiamo preferito utilizzare un record mediamente semplice, ma per i vostri scopi, ovvero la produzione, è importante capire cosa succede quando si passa al .tpl un dato complesso. Per gestirlo all'interno del .tpl la sintassi migliore è la seguente:

```
<table>
{section name="i" loop=$story}
 <tr>
  <td>
    <a href=index.php?modules=
         story&operazione=select&id_story=
       {$story[i].id_story}>{$story[i].id_story}
                                   </a><br>
    <span class=subnav>{$story[i].story} </span>
```

```
    </td>
  </tr>
{/section}
<tr>
    <td>
        <a href=index.php?modules= story&operazione=
            edit&id_story={$story[i].id_story}>Insert
                New Story</a><br>
    </td>
  </tr>
</table>
```

Si tratta di un loop che ci consente di navigare attraverso l'array e stampare a schermo i dati con la formattazione che ci interessa.

INSERIRE I PARAMETRI

È molto probabile che in un blog si voglia consentire a chi legge di cliccare ad esempio sul titolo di una storia per vederne i dettagli, ad esempio la versione estesa. Questo comporta qualche problema, dobbiamo infatti produrre un template separato a seconda delle operazioni che vogliamo fare. Vediamo come cambia il nostro modulo su questa base. Prima di tutto prepariamo all'interno di mysql 5 una nuova stored procedure:

```
DELIMITER $$
DROP PROCEDURE IF EXISTS
    `ioPcms`.`select_one_story`$$
CREATE PROCEDURE `ioPcms`.`select_one_story`
    (idparam INT)
BEGIN
  select * from story where id_story=idparam;
END$$
DELIMITER ;
```

È anche questa abbastanza semplice. Abbiamo solo inserito un parametro da utilizzare all'interno della query. A questo punto dobbiamo gestire il metodo output del modulo story.
Trasformiamolo come segue:

```
<?
include_once('includes/database.php');
class story {
  private $op;
  private $mod;
  private $data=null;
  function story($modules,$operation){
    $this->op=$operation;
    $this->mod=$modules;
    if ($modules == 'story') {
      $this->switchoperation();
    } else {
      $query='call selectstory();';
      $this->data=$this->doquery($query);}}
  private function switchoperation() {
    switch ($this->op) {
      case 'select':
      $query='call select_one_story(
          '.$_GET[id_story].');';
      $this->data=$this->doquery($query);
      break;
      case 'edit':
      $data=array('id_author','story');
      $this->data=$data;
      break;
      case 'store':
      $query="call insertstory(".$_POST[id_author].
          ",'".$_POST[story]."');";
      $this->doquery($query);
      $query='call selectstory();';
      $this->data=$this->doquery($query);
      default:
      $query='call selectstory();';
      $this->data=$this->doquery($query);
      break;}}
  private function doquery($query) {
    $db = new dbase;
    $data = $db->execquery($query);
    return $data;}
  public function operation() {
    return $this->op;}
  public function output() {
    return $this->data;}}
?>
```

Notate che adesso abbiamo fatto un massiccio uso del metodo *switchoperation* e che abbiamo settato anche l'operazione da restituire al template.

INSERIRE UN NUOVO DATO

Fin qui tutto abbastanza semplice. Sia che volessimo visualizzare un elenco di storie, sia che volessimo visualizzare il dettaglio di una sola storia, il nostro template poteva non cambiare. Di fatto era suffciente riempire il tag *{story}* con del testo. Tutto cambia quando invece vogliamo mostrare all'utente una form. Infatti il tag story non può essere riempito con i valori presi dal db, bensì deve restituire al web designer un elenco di nomi corrispondenti a quelli utilizzati all'interno del database. Il Web Designer disegnerà una form e richiamerà il modulo story, dandogli come parametri il nome del modulo, il nome dell'operazione da compiere e i dati in post da inserire nel db. In pratica la form che dovrà comparire nel template sarà qualcosa del genere:

```
<form action=index.php?modules=
```

```
                story&operazione=store method="POST">
{section name="i" loop=$story}
<b>{$story[i]}:</b><input type=text name=
                                  {$story[i]}><br>
{/section}
<input type=submit>
</form>
```

quando l'utente cliccherà sul submit, tramite il metodo *"switchoperation"* del modulo (do per scontato che abbiate capito come l'engine instanzia un oggetto di tipo *story*), succederà qualcosa del genere:

```
case 'store':
   $query="call insertstory(".$_POST[id_author].
                         ",'".$_POST[story]."');";
   $this->doquery($query);
```

la stored procedure che effettua l'insert è la seguente:

```
DELIMITER $$
DROP PROCEDURE IF EXISTS `ioPcms`.`insertstory`$$
CREATE PROCEDURE `insertstory`(idauthor
                                INT,storia TEXT)
BEGIN
   insert into story (id_autore,story) values
                              (idauthor,storia);
END$$
DELIMITER ;
```

FACCIAMO UN PASSO INDIETRO

Per poter visualizzare la form di inserimento, l'utente avrà dovuto inviare al modulo il parametro edit. Avrete notato che nel template esisteva qualcosa del genere:

```
<a href=index.php?modules=story&operazione=
                 edit&id_story={$story[i].id_story}>
                             Insert New Story</a><br
```

che appunto richiamava con un link il modulo *'story'*, passandogli il parametro 'edit'. Anche in questo caso il metodo *switchoperation* conteneva:

```
         case 'edit':
                 $data=array('id_author','story');
                 $this->data=$data;
         break;
```

Ora, non ci siamo ancora occupati dei template condizionali. Se avete seguito tutto, saprete anche che al template abbiamo passato anche un parametro "modulo+operation", ad esempio se il modulo è story, e l'operation è edit, il template potrà usare un tag "storyoperation" ed è esattamente quello che sfrutteremo noi per gestire i template condizionali come segue:

```
{if $storyoperation neq "edit"}
<table>
{section name="i" loop=$story}
  <tr>
    <td>
       <a href=index.php?modules=story&operazione=
                select&id_story={$story[i].id_story}
                     >{$story[i].id_story}</a><br>
       <span class=subnav>{$story[i].story} </span>
    </td>
  </tr>
{/section}
<tr>
    <td>
       <a href=index.php?modules=story&operazione=
                edit&id_story={$story[i].id_story}>
                           Insert New Story</a><br>
    </td>
  </tr>
</table>
{else}
<form action=index.php?modules=
           story&operazione=store method="POST">
{section name="i" loop=$story}
<b>{$story[i]}:</b><input type=text name=
                                 {$story[i]}><br>
{/section}
<input type=submit>
</form>
{/if}
```

CONCLUSIONI

La struttura che vi abbiamo illustrato è una delle tante possibili per la creazione di un engine per la costruzione di CMS skinnable e modulari. Certamente può essere raffinata e migliorata per le vostre esigenze. L'importante è essere riusciti a capire come creare un'architettura che divida in tre parti ben distinte il progetto. Alcuni pezzi di codice presentati in questo articolo potranno risultare particolarmente difficili da comprendere per i neofiti del PHP, ma se avrete la pazienza di rileggere più volte l'articolo, scoprirete alcuni passaggi molto utili ed eleganti, come ad esempio quello utilizzato nel metodo "factory" che non a caso rappresenta il punto nevralgico del sistema. Inoltre il massiccio uso della programmazione ad oggetti e l'introduzione alle stored procedure di MySQL rappresenta un primo passo verso una logica nuova di programmazione, a cui i programmatori PHP devono tendere se vogliono far compiere a questo linguaggio un nuovo importante salto di qualità.

PHP

COME POSSO AGGIUNGERE UN'APPLICAZIONE NELLA TRAYBAR?

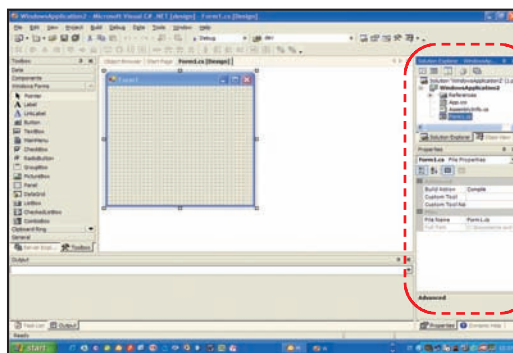
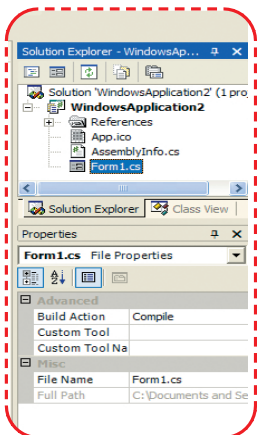
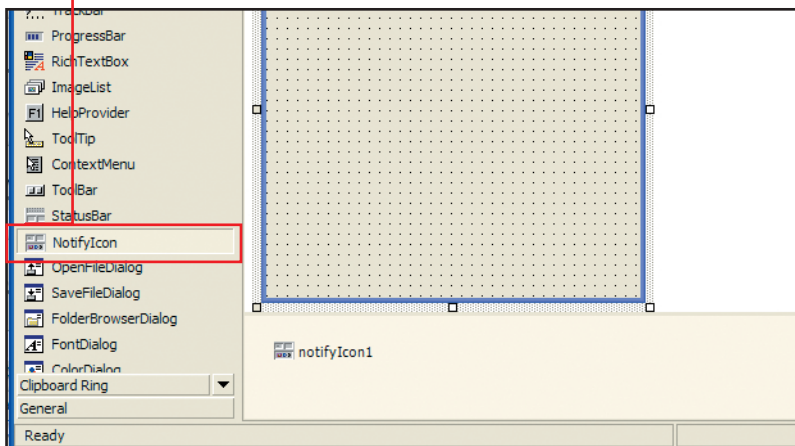
LA SYSTEM TRAY È L'AREA DELLA BARRA DI WINDOWS (NELL'ANGOLO IN BASSO A DESTRA DELLO SCHERMO) CHE COMPRENDE AD ESEMPIO L'OROLOGIO DI SISTEMA. VEDIAMO COME USARLA

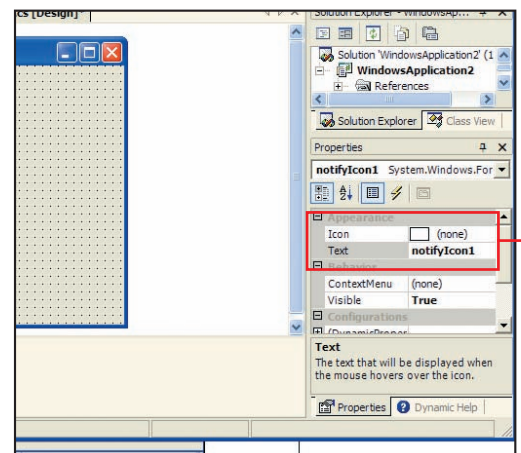
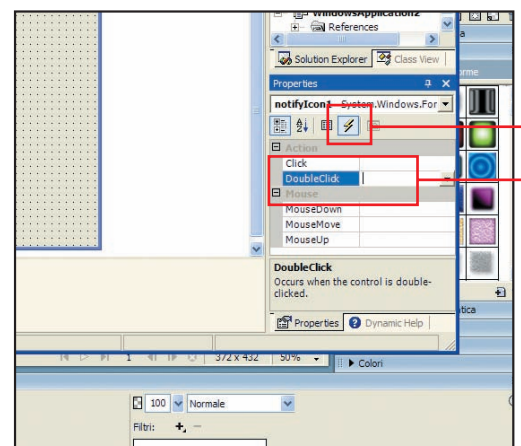
C#

VISUAL BASIC.NET

Trovano posto in quest'area programmi che devono girare in background e che possono essere massimizzati per eseguire un eventuale settaggio o visualizzare risultati specifici. Esempi tipici sono i software d'antivirus o i firewall. Normalmente le applicazioni vengono massimizzate allo stato naturale cliccando due volte sull'icona nella traybar che li rappresenta.

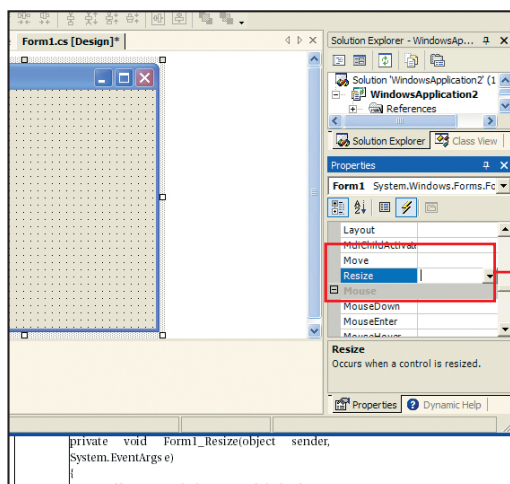
FACCIAMOLO IN C#

1 Posizioniamoci sulla Windows Form C# principale del programma

2 Dalla toolbox sulla sinistra della form selezioniamo il controllo *NotifyIcon* e posizioniamolo sulla form

Nella parte bassa sotto la form comparirà un etichetta che indica che il componente *NotifyIcon* è ora utilizzabile all'interno del nostro progetto.

3 Selezioniamo il componente appena creato e nella finestra delle proprietà assegnamo l'*icona* che dovrà contraddistinguere il programma nella tray bar e un nome per l'applicazione

4 Nella sezione *Events* alla voce *DoubleClick* eseguiamo un doppio click con il mouse

5 In seguito alla doppia pressione del tasto sull'evento double click verrà generato il template di gestione dell'evento. Qui dentro scriveremo il codice

```
private void notifyIcon1_DoubleClick(object sender,
                    System.EventArgs e)
```

```
{
    Show();
   WindowState = FormWindowState.Normal;
}
```

6 Selezionate la form e questa volta nella tabella degli eventi selezionate l'evento *resize*.

Come al passo 5 cliccando due volte verrà generato il codice di gestione dell'evento, che potremo personalizzare come segue:

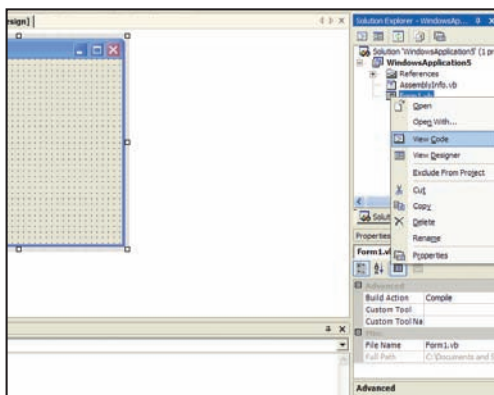
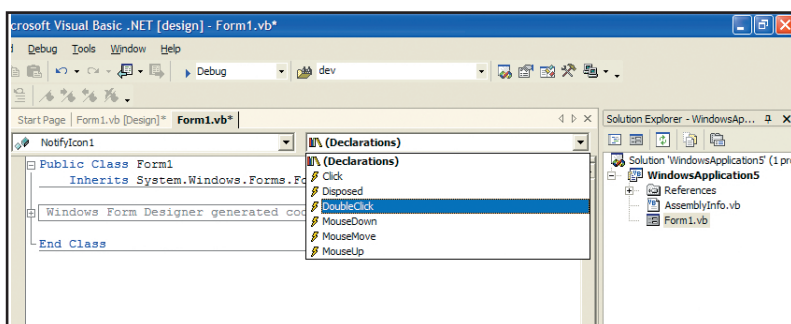
```
private void Form1_Resize(object sender,
                         System.EventArgs e)
{
   if (FormWindowState.Minimized == WindowState)
      Hide();
}
```

COME FUNZIONA?

Quando l'utente riduce a icona l'applicazione si scatena un evento di tipo "Resize", che viene gestito dal nostro codice. In sostanza molto semplicemente l'applicazione viene nascosta alla vista. Rimane visibile solo la sua rappresentazione sulla tray bar.
Allo stesso modo quando l'utente clicca due volte sull'icona nella traybar viene generato un evento double click e gestito dal nostro codice, che non fa altro che rendere visibile la form con il comando show e fissare la sua dimensione a normal.

REALIZZIAMOLO IN VB.NET

Fino al passo 4 il procedimento è del tutto identico. Per gestire gli eventi invece è necessario

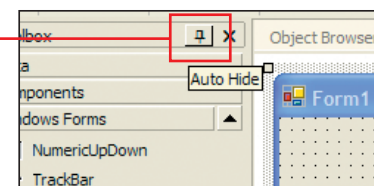
1 Portarsi in *"Solution Explorer"* a destra nello schermo e cliccare con il tasto destro del mouse. Dal menu a discesa che comparirà selezionare *"View Code"*

2 Dalla casellina a discesa sulla sinistra selezionare *"NotifyIcon"* mentre dalla casellina degli eventi selezionare *"DoubleClick"*2

3 Il codice da inserire in relazione all'evento *"DoubleClick* è il seguente:

```
Private Sub NotifyIcon1_DoubleClick(ByVal sender
    As Object, ByVal e As System.EventArgs) Handles
                              NotifyIcon1.DoubleClick
       Show()
       WindowState = FormWindowState.Normal
   End Sub
```

Per l'evento *resize* invece il codice è il seguente:

```
  Private Sub Form1_Resize(ByVal sender As Object,
             ByVal e As System.EventArgs) Handles
                                     MyBase.Resize
       If FormWindowState.Minimized =
                                 WindowState Then
           Hide()
       End If
End Sub
```

SE LA FINESTRA DEGLI STRUMENTI SCOMPARE

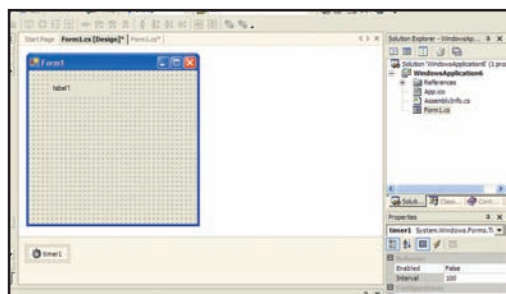
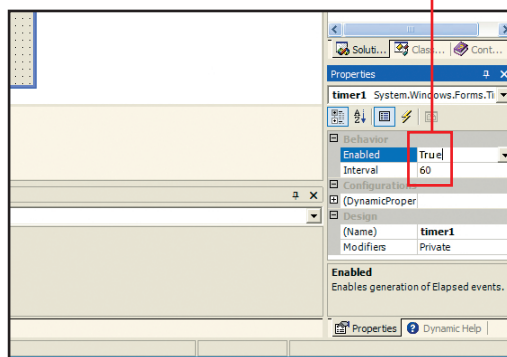
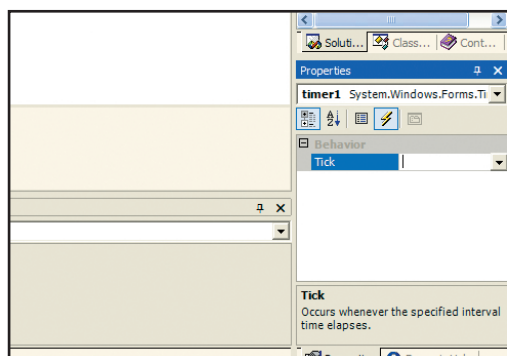
Normalmente la *"toolbox"*, cioè la finestra degli strumenti di Visual Studio, scompare dopo avera utilizzata. Per renderla permanente sullo schermo è possibile cliccare sull'icona *"chiodino"* che la fissa allo schermo.

COME POSSO ESEGUIRE UNO SPEZZONE DI CODICE OGNI MINUTO?

SI PUÒ UTILIZZARE UN COMPONENTE DI TIPO TIMER CHE NASCE PER RISOLVERE QUESTO TIPO DI PROBLEMA

C#

VISUAL BASIC.NET

FACCIAMOLO IN C#

1 Dalla toolboox trasciniamo nella form un componente di tipo *"timer"* e una label

2 Dalla finestra delle proprietà del timer poniamo *Enabled = True* e *Interval = 60*

3 Nella finestra degli eventi clicchiamo due volte sull'evento *"Tick"*.

4 Il codice da inserire è il seguente:

```
if(sender==Clock)
  {
    lbTime.Text=GetTime();
  }
```

5 Non ci rimane che definire la procedura *GetTime()*;

```
public string GetTime()
  {
    string TimeInString="";
    int hour=DateTime.Now.Hour;
    int min=DateTime.Now.Minute;
    int sec=DateTime.Now.Second;
    TimeInString=(hour < 10)?"0" + hour.ToString()
                                      :hour.ToString();
    TimeInString+=":" + ((min<10)?"0" +
                          min.ToString() :min.ToString());
    TimeInString+=":" + ((sec<10)?"0" +
                          sec.ToString() :sec.ToString());
    return TimeInString;
  }
```

SINTASSI SEMPLIFICATA

Ad alcuni potrebbe non risultare familiare la notazione:

```
TimeInString=(hour <
      10)?"0" +
   hour.ToString()
   :hour.ToString();
```

Si tratta di un if abbreviato che consente di inserire uno 0 davanti all'ora se è minore di 10. Corrisponde a:

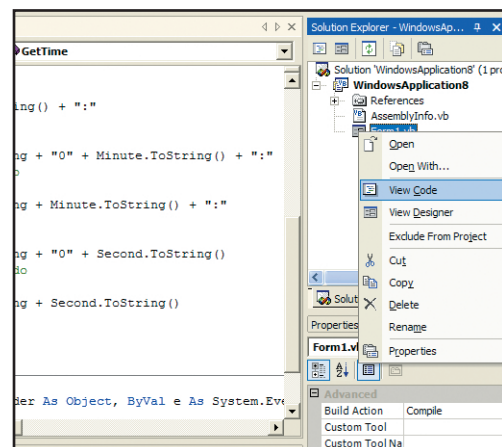
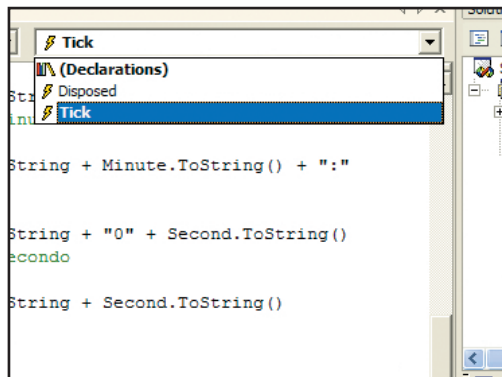
```
if (hour<10)
  TimeInString="0"+
        hour.ToString();
        //mette 0 davanti
              all'ora
else
  TimeInString=hour
          .ToString();
```

COME FUNZIONA?

Il timer genera un evento tick ogni 60 millisecondi. In relazione all'evento viene richiamato il metodo GetTime che prende l'orario di sistema scomposto in tre interi tramite i metodi Datetime.Now.. e ricostruisce una stringa che viene utilizzata per visualizzare l'orario corrente

FACCIAMOLO IN VB.NET

Fino al passo 3 la procedura è identica.

1 Per inserire il codice, portatevi nel *"Solution Explorer"* e con il tasto destro cliccate sulla form. Nel menu che segue cliccate su *"view code"*.

2 Nella pagina successiva, selezionate dal menu di sinistra *"timer1"* e in quello di sinistra *"Tick"*.

3 Il codice da inserire è il seguente:

```
Private Sub Timer1_Tick(ByVal sender As Object,
  ByVal e As System.EventArgs) Handles Timer1.Tick
      Label1.Text = GetTime()
End Sub
```

4 La funzione *GetTime()* può essere inserita poco prima della *Timer1_Tick()* ed avrà la seguente forma:

```
    Private Function GetTime() As String
        Dim TimeInString As String
        Dim Hour As Int16
        Dim Minute As Int16
        Dim Second As Int16
        Hour = DateTime.Now.Hour
        Minute = DateTime.Now.Minute
        Second = DateTime.Now.Second

        If (Hour < 10) Then
            TimeInString = "0" + Hour.ToString() + ":"
            'mette 0 davanti all'ora
        Else
            TimeInString = Hour.ToString() + ":"
        End If
        If (Minute < 10) Then
            TimeInString = TimeInString + "0" +
                                     Minute.ToString() + ":"
            'mette 0 davanti al Minuto
        Else
            TimeInString = TimeInString +
                                    Minute.ToString() + ":"
        End If
        If (Second < 10) Then
            TimeInString = TimeInString + "0" +
                                         Second.ToString()
            'mette 0 davanti al Secondo
        Else
            TimeInString = TimeInString +
                                         Second.ToString()
        End If
        Return TimeInString
    End Function
```

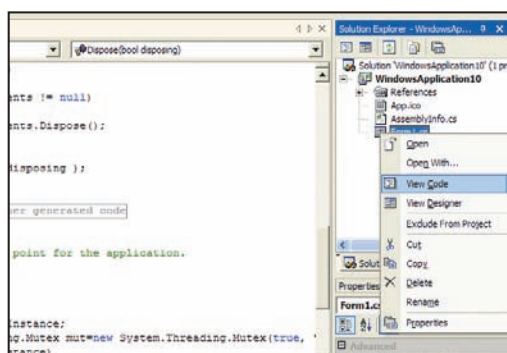
ESEGUIRE UNA SINGOLA ISTANZA DELL'APPLICAZIONE

PER DEFAULT, DI UNA SINGOLA APPLICAZIONE È POSSIBILE ESEGUIRE PIÙ ISTANZE SU UNO STESSO PC: CI SONO DEI CASI IN CUI È NECESSARIO LIMITARNE L'USO AD SINGOLA ESECUZIONE. IN C# E VISUAL BASIC, LA COSA È SEMPLICISSIMA AGENDO SUL METODO MAIN() DELL'APPLICAZIONE UTILIZZANDO UN OGGETTO SYSTEM.THREADING.MUTEX

C#

VISUAL BASIC.NET

FACCIAMOLO IN C#

1 Con il tasto destro del mouse clicchiamo nel *Solution Explorer* su *"View Code"*

2 Ricerchiamo all'interno del codice il metodo *Main* e inseriamo il codice che segue:

```
using System.Threading;
...
static void Main()
{
   bool bAppFirstInstance;
  System.Threading.Mutex mut=new
           System.Threading.Mutex(true, "Global\\" +
                "APP_NAME", out bAppFirstInstance);
  if(bAppFirstInstance)
Application.Run(new Form1());
  else
      MessageBox.Show("Esiste già un'esecuzione in
```

```
corso!", "Startup warning",
    MessageBoxButtons.OK,
        MessageBoxIcon.Exclamation,
        MessageBoxDefaultButton.Button1);
}
```

COME FUNZIONA

Viene dichiarata una variabile Booleana "bAppFirsInstance". Viene poi inizializzato un oggetto "mut" di classe "System .Threading .Mutex" al quale nel costruttore vengono passati tre parametri: un valore booleano, fissato a true indica che il thread che chiama il mutex ne è anche il possessore, una stringa che indica il nome del mutex, ed una variabile che contiene un valore booleano. Quest'ultima variabile varrà true se il thread che richiama il mutex ne è anche il possessore.
La prima volta che l'applicazione viene eseguita, il thread che chiama il mutex ne diventa anche proprietario, perciò bAutFirstInstance varrà true. La seconda volta il thread non riuscirà a diventare proprietario del mutex, di fatto c'è già un proprietario, ed è per questo che la variabile "bAutFirstInstance" varrà "False". Controllando questa variabile sarà possibile eseguire l'applicazione, oppure generare una msgbox che avverte che l'applicazione è già in esecuzione.

FACCIAMOLO IN VB.NET

Il passo 1 rimane sostanzialmente identico. In Visual Basic.NET però non troverete la funzione *Main* precostruita come in C#, dovrete riscriverla come segue e posizionarla poco al di sotto dell'etichetta *#End Region*

```
Public Shared Sub Main()
    Dim bAppFirstInstance As Boolean
    Dim mut As New System.Threading.Mutex(True,
                    "APP_NAME", bAppFirstInstance)
    If (bAppFirstInstance) Then
        Application.Run(New Form1)
    Else
        MessageBox.Show("Esiste già un'esecuzione in
        corso!", "Startup warning", MessageBoxButtons
                .OK, MessageBoxIcon.Exclamation,
                MessageBoxDefaultButton.Button1)
    End If

End Sub
```

COME POSSO CONTROLLARE CHE UN SITO STIA FUNZIONANDO?

CAPITA MOLTO SPESSO DI AVERE LA NECESSITÀ DI TESTARE LA RAGGIUNGIBILITÀ DI UN INDIRIZZO WEB SU UNA RETE. IN SOFTWARE DI TIPO CLIENT /SERVER, POTREBBE ESSERE NECESSARIO PER TESTARE LA CONNESSIONE PRIMA DI AVVIARE UNA PROCEDURA. IN QUESTO TIP VEDREMO COME EFFETTUARE UN PING CON POCHE ISTRUZIONI CHE CI RITORNI UN TRUE SE L'INDIRIZZO È RAGGIUNGIBILE, ALTRIMENTI UN FALSE

C#

VISUAL BASIC.NET

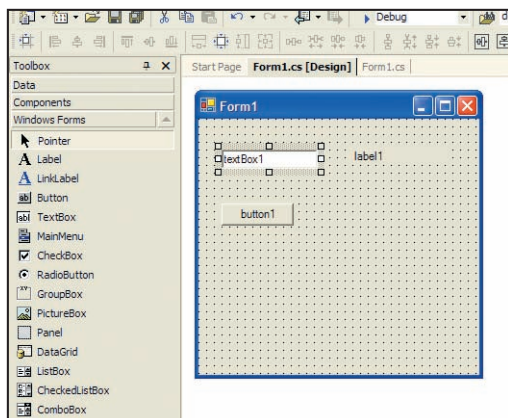
I NAMESPACE DA IMPORTARE

Perché tutto funzioni è necessario importare il namespace *System.Net.Sockets*, per farlo è sufficiente aggiungere le righe

```
Using System.Net
Using System.Net.Sockets
```

in alto sotto gli altri import utilizzati nell'applicazione.

FACCIAMOLO IN C#

1 Posizioniamoci sulla Windows Form C# principale del programma e posizioniamo su di essa dalla toolbox, una label, un textbox e un bottone

2 Clicchiamo due volte sul bottone per creare il template del codice che verrà eseguito in relazione all'evento *"click"*. Il codice da inserire è il seguente:

```
private void button1_Click(object sender,
                    System.EventArgs e)
  {
        bool alive;
        alive = pingUrl(10,textBox1.Text);
        if (alive == true )
          label1.Text="Ok";
        else  label1.Text="Ko";
}
```

3 Non resta che definire il metodo *pingUrl* come segue

```
private static bool pingUrl(int retries, string URL)
```

```
{
    if ( retries == 0 ) throw new
    ArgumentOutOfRangeException("retries", retries,
            "Il valore non può essere minore di 1");
    bool connected = false;
  TcpClient client = new TcpClient();
  try
  {
        client.Connect(URL, 80);
        client.Close();
        connected = true;
    }
    catch (SocketException)
    {
        connected = false;
    }
    return connected;
}
```

COME FUNZIONA

Alla pressione del bottone si scatena un evento "click" che viene gestito dal metodo button1_click. In questo metodo viene definita una variabile booleana boolAlive che assume un valore ritornato dal metodo pingUrl. A tale metodo vengono passati un numero intero che indica il numero di ping da effettuare prima di ottenere una risposta e un valore di testo prelevato dalla casella textbox1, che contiene l'indirizzo da pingare. Il metodo in questione instanzia un oggetto Client di tipo TcpCliente che fisicamente si occupa di mandare un ping al sito indicato. Se il ping da esito positivo viene ritornato un valore true, altrimenti false.

FACCIAMOLO IN VB.NET

Fino al punto 2 i passi sono identici. Il codice da inserire nell'evento di gestione del pulsante è il seguente:

```
Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                                Handles Button1.Click
        Dim alive As Boolean
        alive = pingUrl(10, TextBox1.Text)
        If (alive = True) Then
            Label1.Text = "Ok"
        Else Label1.Text = "Ko"
        End If
```

Il metodo *pingUrl* sarà invece strutturato come segue:

```
Private Function pingUrl(ByVal retries As Int16, ByVal
                                        URL As String)
        If (retries = 0) Then
            Throw New ArgumentOutOfRangeException(
             "retries", retries, "Il valore non può essere
                                          minore di 1")
        End If
        Dim connected As Boolean = False
        Dim client As New System.Net.Sockets.TcpClient
        Try
            client.Connect(URL, 80)
            client.Close()
            connected = True
        Catch ex As SocketException
            connected = False
        End Try
        Return connected

    End Function
```

Come sempre potete scrivere questo codice immediatamente prima di quello di gestione dell'evento *click*.

I NAMESPACE PER VISUAL BASIC

Anche in questo caso perché tutto funzioni è necessario importare il namespace System.Net.Sockets, per farlo è sufficiente aggiungere le righe

```
Imports System.Net
Imports System.Net
                    .Sockets
```

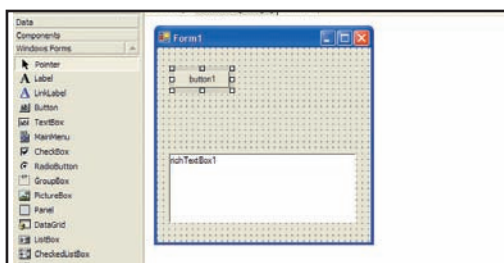
in alto poco prima della dichiarazione della classe Form1.

COME OTTENERE L'ELENCO DEGLI INDIRIZZI IP DI UNA MACCHINA?

SEMPRE PIÙ FREQUENTEMENTE ACCADE CHE UNA MACCHINA POSSA ESSERE ATTREZZATA CON PIÙ DI UN INDIRIZZO DI RETE. VEDIAMO COME RECUPERARE UN ELENCO DEGLI INDIRIZZI ATTRIBUITI AD UN PC

FACCIAMOLO IN C#

1 Posizioniamoci sulla Windows Form C# principale del programma e posizioniamo su di essa dalla toolbox, un bottone e una richtextbox. Potete anche cambiare la caption del bottone per rendere l'interfaccia più attraente.
Allo stesso modo potete cambiare la caption sulla form.

C#

VISUAL BASIC.NET

I NAMESPACE DA USARE IN C#

Perché tutto funzioni è necessario importare il namespace System.Net .Sockets, per farlo è sufficiente aggiungere le righe

```
Using System.Net
Using System.Net.Sockets
```

in alto sotto gli altri import utilizzati nell'applicazione

I NAMESPACE DA IMPORTARE IN VB .NET

È necessario importare due namespace per la gestione della rete. In particolare

```
Imports System.Net
Imports System.Net
.Sockets
```

È sufficiente scrivere queste due righe in alto poco prima della dichiarazione della classe Form1.

2 Clicchiamo due volte sul bottone per creare il template del codice che verrà eseguito in relazione all'evento "*click*".
Il codice da inserire è il seguente:

```
private void button1_Click(object sender,
                              System.EventArgs e)
{  richTextBox1.Text=getIpList(); }
```

3 Dovremo definire il metodo *getIpList*. Il codice può essere inserito poco prima del metodo che gestisce l'evento del bottone ed è il seguente

```
public static string getIpList()
  {  String strHostName = Dns.GetHostName();
     String buffer = "";
     IPHostEntry iphostentry = Dns.GetHostByName(
                                   strHostName);
     foreach(IPAddress ipaddress in
                           iphostentry.AddressList)
     {  buffer += ipaddress.ToString()+"\n";  }
      return buffer;
  }
```

COME FUNZIONA

Alla pressione del bottone si scatena un evento "click" che viene gestito dal metodo button1_click. L'evento richiama semplicemente un metodo getIpList che provvede a restituire una stringa che contiene l'elenco degli indirizzi IP e che riempirà la richTextBox. La getIpList ricava il nome del pc su cui stiamo eseguendo il programma tramite il metodo Dns.GetHostname e inizializza un buffer come stringa vuota. In un secondo momento dichiara un oggetto i classe IPHostEntry e lo riempie con gli indirizzi ip prelevati dal metodo Dns.GetHostByName. Il successivo ciclo, semplicemente provvede a costruire la stringa da restituire.

FACCIAMOLO IN VB.NET

Fino al punto 2 i passi sono identici. Il codice da inserire nell'evento di gestione del pulsante è il seguente:

```
Private Sub Button1_Click(ByVal sender As
                          System.Object, ByVal e
      As System.EventArgs) Handles Button1.Click
      RichTextBox1.Text = getIpList()
  End Sub
```

Il metodo getIpList() sarà invece strutturato come segue:

```
Private Function getIpList() As String
      Dim strHostName As String =
                               Dns.GetHostName()
      Dim buffer As String = ""
      Dim iphostentry As IPHostEntry =
                  Dns.GetHostByName(strHostName)
      For Each ipaddress As IPAddress In
                         iphostentry.AddressList()
         buffer += ipaddress.ToString() + Chr(13)
      Next
      Return buffer

  End Function
```

Come sempre potete scrivere questo codice immediatamente prima di quello di gestione dell'evento click.

CODICE PIÙ SINTETICO

La sintassi abbreviata

```
buffer += ipaddress.ToString() + Chr(13)
```

Corrisponde a

```
buffer = buffer + ipadress.ToString() + chr(13)
```

RIDEFINIRE OPERATORI IN C#

Il C# permette al programmatore di definire, o ridefinire, il significato degli operatori (come &, |, *, +, ecc) all'interno delle proprie classi, originando così delle funzionalità personalizzate. In questo modo non è strettamente necessario richiamare un metodo per eseguire un'operazione.
Ad esempio, possiamo definire un nostro ArrayList che consente di aggiungere un record alla collection usando l'operatore + oppure di concatenare due collection usando lo stesso operatore

```
class MyArray: System.Collections.ArrayList
{
  public static MyArray operator +
                          (MyArray ar, Object ob)
  {
      ar.Add(ob);
      return ar;
  }
  public static MyArray operator +
                     (MyArray ar1, MyArray ar2)
  {
      ar1.AddRange(ar2);
        return ar1;
  }
}
```

In questo modo possiamo fare:

```
MyArray m1=new MayArray();
MyArray m2=new MayArray();
String s1="pippo";
String s2="pluto";
m1+=s1; m2+=s2;
MyArray m3=m1+m2;
```

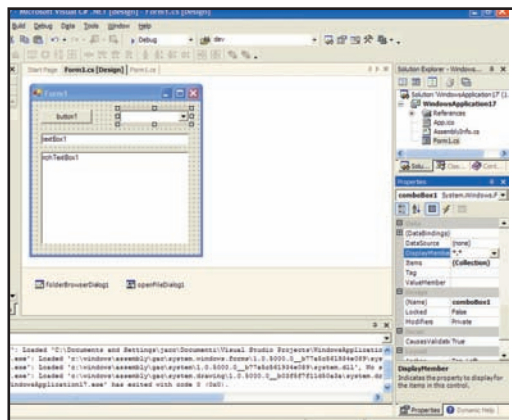
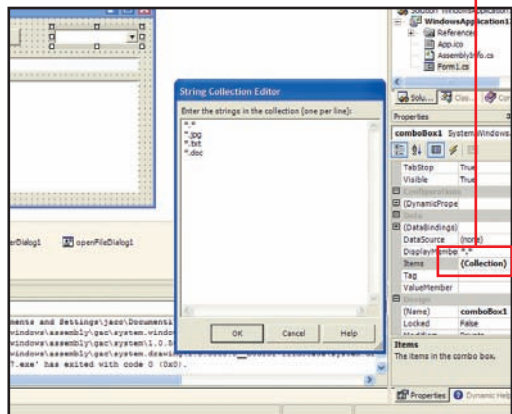
COME POSSO ELENCARE IL CONTENUTO DI UNA DIRECTORY?

UN METODO UTILE È QUELLO CHE UNISCE ANCHE UN FILTRO PER LA VISUALIZZAZIONE SULLA BASE DELL'ESTENSIONE DEL FILE. IN QUESTO MODO SI POSSONO MOSTRARE AD ESEMPIO TUTTI I FILE *.RTF PIUTTOSTO CHE *.DOC ETC, MODIFICANDO SEMPLICEMENTE IL CONTENUTO DEL FILTRO

C#

VISUAL BASIC.NET

FACCIAMOLO IN C#

1 Posizioniamoci sulla Windows Form C# principale del programma e trasciniamo su di essa dalla toolbox, un bottone, una richtextbox, un combobox, e una folderbrowserdialog

2 Selezioniamo il combobox e di seguito nelle proprietà scegliamo items. Cliccando sul simbolo [...] sul lato destro comparirà una finestra, sarà qui dentro che scriveremo un elenco di filtri possibili

3 Clicchiamo due volte sul bottone per ottenere il template del codice di gestione dell'evento click. Il codice da inserire è il seguente

```
private void button1_Click(object sender,
                                    System.EventArgs e)
{
    string filtro = comboBox1.SelectedItem.ToString();
    folderBrowserDialog1.ShowDialog();
    textBox1.Text = folderBrowserDialog1
                              .SelectedPath.ToString();
    richTextBox1.Text = ListFiles(textBox1.Text,filtro);
}
```

4 A questo punto possiamo definire il metodo *ListFiles* che provvederà a recuperare l'elenco dei file e riempire la RichTextBox.
Il codice è il seguente:

```
private string ListFiles(string path, string filtro)
{
   string buffer ="";
   foreach (string files in System.IO.Directory.GetFiles(
                                          path, filtro))
   {
      buffer += files+"\n";
    }
   return buffer;
}
```

COME FUNZIONA

Il meccanismo è abbastanza semplice.
Tutto viene delegato al metodo ListFiles, che non fa altro che eseguire un ciclo sulla collection riempita da *System.IO.Directory.GetFiles*(path, filtro) e restituire un a stringa contenente l'elenco voluto. Le altre istruzioni servono per recuperare il path su cui eseguire il comando e per settare un eventuale filtro sui file che si intendono ottenere.

FACCIAMOLO IN VB.NET

Fino al punto 3 i passi sono identici. Il codice da inserire nell'evento di gestione del pulsante è il seguente:

```
Private Function ListFiles(ByVal path As String,
                              ByVal filtro As String) As String
     Dim Buffer As String = ""
     For Each files As String In
          System.IO.Directory.GetFiles(path, filtro)
     Buffer += files + Chr(13)
     Next
     Return Buffer

End Function
```

Il metodo ListFiles() sarà invece strutturato come segue:

```
Private Function getIpList() As String
    Dim strHostName As String = Dns.GetHostName()
    Dim buffer As String = ""
    Dim iphostentry As IPHostEntry =
                         Dns.GetHostByName(strHostName)
    For Each ipaddress As IPAddress In
                              iphostentry.AddressList()
        buffer += ipaddress.ToString() + Chr(13)
    Next
    Return buffer
End Function
```

Come sempre potete scrivere questo codice immediatamente prima di quello di gestione dell'evento click.

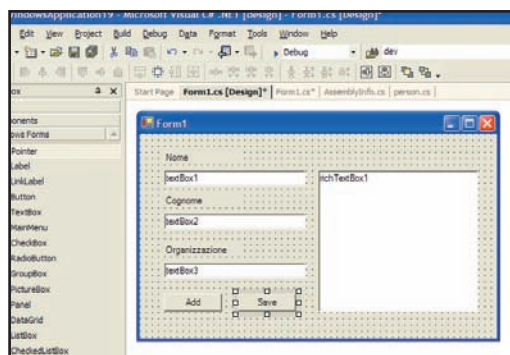
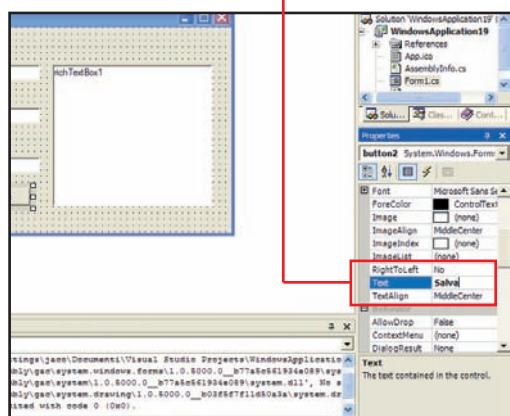
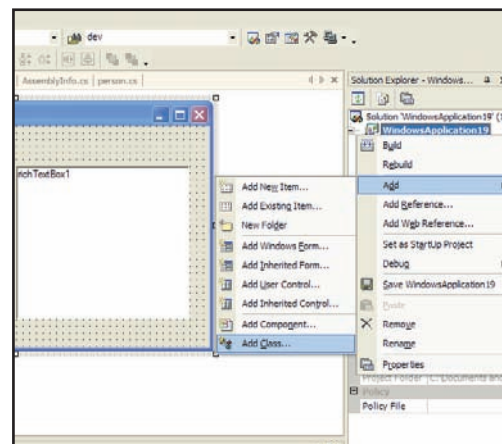
CHE COSA VUOL DIRE SERIALIZZARE?

LA SERIALIZZAZIONE È UN'OPERAZIONE CHE CONSENTE DI MEMORIZZARE LO STATO DI UN OGGETTO PER POI RECUPERARLO IN UN SECONDO MOMENTO. QUANDO PARLIAMO DI SALVARE LO STATO DI UN OGGETTO INTENDIAMO DIRE L'INSIEME DELLE PROPRIETÀ CHE LO CONTRADDISTINGUONO, OVVERO LA RAPPRESENTAZIONE BINARIA (O IN ALTRO FORMATO) DI UN OGGETTO INTERO.

C#

VISUAL BASIC.NET

FACCIAMOLO IN C#

1 Posizioniamoci sulla Windows Form C# principale del programma e posizioniamo su di essa dalla toolbox, tre TextBox, una richtextbox, e due bottoni. Aggiungiamo anche qualche label informativa, "Nome, Cognome, Organizzazione"

2 Assicuriamoci di assegnare ai due bottoni le corrispondenti etichette "Add" e "Salva". Per farlo selezioniamo i bottoni uno alla volta e agiamo sulla proprietà *Text*, dalla finestra delle proprietà

3 Clicchiamo con il tasto destro del mouse nel "Solution Explorer" sulla voce che rappresenta la nostra soluzione. Dal menu a tendina scegliamo "Add" e poi "Add Class".
Nella successiva finestra digitiamo "person" e clicchiamo su ok.

4 Grazie al passo precedente verrà creato lo scheletro del codice di un nuova classe, appunto la classe person.
La nostra nuova classe sarà composta come segue:

```
using System;
namespace WindowsApplication19
{
  /// <summary>
  /// Summary description for person.
  /// </summary>
  ///
  [Serializable]
  public class person
  {
```

GLI IMPORT DA ESEGUIRE PER VB .NET

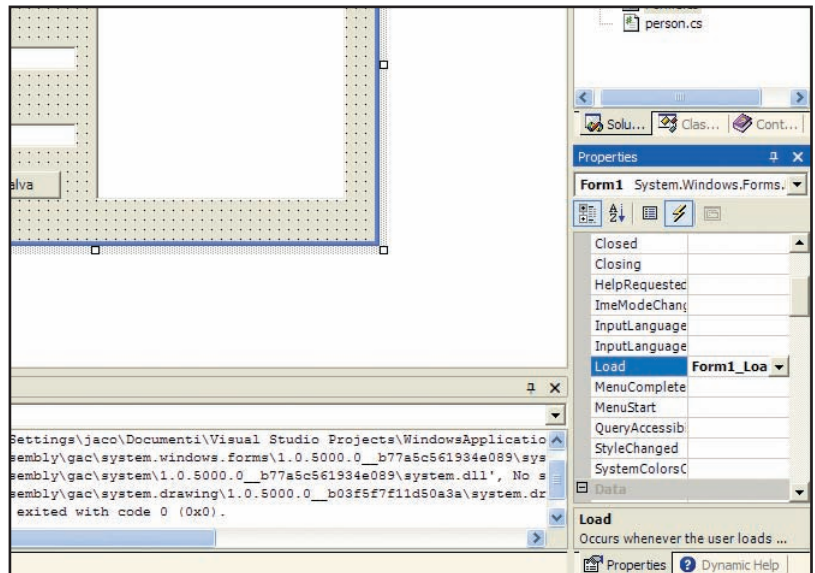
Sono:

```
Imports System.Runtime
          .Serialization
Imports System.Runtime
   .Serialization.Formatter
Imports System.IO
```

```
        private string nome;
        private string cognome;
        private string organizzazione;
        public person()
        {

        }
        public string Cognome
        {
            get
            {
                return cognome;
            }
            set
            {
                cognome=value;
            }
        }
        public string Nome
        {
            get
            {
                return nome;
            }
            set
            {
                nome=value;
            }
        }
        public string Organizzazione
        {
            get
            {
                return organizzazione;
            }
            set
            {
                organizzazione=value;
            }
        }
    }
}
```

5 Torniamo alla finestra di disegno della form e clicchiamo due volte sul bottone “Add” per creare il template di gestione dell’evento “OnClick”. Il nostro codice sarà il seguente:

```
private void button1_Click(object sender,
                           System.EventArgs e)
{
   persona[indice] = new person();
   persona[indice].Nome=textBox1.Text;
   persona[indice].Cognome=textBox2.Text;
   persona[indice].Organizzazione=textBox3.Text;
   richTextBox1.Text+=persona[indice].Nome+"
                     "+persona[indice].Cognome+"
                  "+persona[indice].Organizzazione;
                                          indice++;

}
```

6 Torniamo alla finestra di disegno della form, selezioniamola, agiamo sul pulsante che ci consente di passare alla gestione degli eventi e clicchiamo due volte sull’evento Load per creare il codice che verrà eseguito nel momento in cui la form verrà caricata e sarà disponibile all’utente

7 Il codice da inserire e che verrà eseguito al caricamento della form è il seguente:

```
private void Form1_Load(object sender,
                        System.EventArgs e)
 {
      System.IO.StreamReader reader = null;
      richTextBox1.Text="";
      try
      {
         stem.Runtime.Serialization.Formatters.Binary
                    .BinaryFormatter bin = new
                    System.Runtime.Serialization
              .Formatters.Binary.BinaryFormatter();
         reader = new System.IO.StreamReader(
                Environment.CurrentDirectory+@
                                "\persona.bin");
         persona = (person[])bin.Deserialize(
                             reader.BaseStream);
         reader.Close();
         foreach (person dato in persona)
         {
            richTextBox1.Text+=dato.Nome+"
                         "+dato.Cognome+"\n";
         }
      }
      catch
      {
```

```
        }
        finally
        {
            if (reader != null ) reader.Close();
        }

    }
}
```

COSA VUOL DIRE SERIALIZABLE?

L'etichetta Serializable indica appunto che la classe che segue è serializzabile

8 Torniamo alla pagina di disegno della form e clicchiamo due volte sul bottone "Salva" per generare il template di gestione dell'evento *clicl* su quel bottone.
Il codice da inserire è il seguente:

```
private void button2_Click(object sender,
                           System.EventArgs e)
{
    System.Runtime.Serialization.Formatters
                   .Binary.BinaryFormatter
    bin = new System.Runtime.Serialization.Formatters
                   .Binary.BinaryFormatter();
    System.IO.StreamWriter dat = new
                   System.IO.StreamWriter(
    Environment.CurrentDirectory+@"\persona.bin");
    bin.Serialize(dat.BaseStream, persona);
    dat.Close();
}
```

GLI IMPORT DA AGGIUNGERE IN C#

Perché tutto funzioni è necessario aggiungere gli import

```
using System.Data;
using System.IO;
using System.Runtime.Serialization;
using System.Runtime
        .Serialization.Formatters.Binary;
```

in alto nel codice nella sezione che contiene tutti gli altri. In alto nella dichiarazione della classe è necessario aggiungere invece:

```
private int indice;
private person[] persona =
                  new person[10];
```

COME FUNZIONA

Abbiamo creato una nuova classe "Persona" contenente proprietà e metodi che ci consentono di instanziare degli oggetti Persona dotati di nome, cognome, organizzazzione.
Con la dichiarazione iniziale private person[] persona = new person[10]; creiamo un array di oggetti di classe persona. Alla pressione del tasto add creiamo un oggetto person appartenente all'array. Per creare i vari campi dell'array utilizziamo un indice dichiarato a livello globale nella classe.
Associamo i vari campi nome e cognome all'oggetto persona[x] e riempiamo il richtextbox laterale con gli stessi campi. In sostanza al termine avremo qualcosa del genere:

```
persona[0].Nome="Pippo"
persona[0].Cognome="Pluto"
persona[1].Nome="Paolino"
persona[1].Nome="Paperino"
```

Alla chiusura della form ci aspetteremmo che gli oggetti in questione vengano distrutti, per cui alla successiva riapertura il richtextbox sarebbe vuoto e anche i nostri dati persi. Possiamo evitare questa situazione tramite il tasto save. Azionando questo tasto viene instanziato un oggetto bin di classe System.Runtime.Serialization.Formatters .Binary.BinaryFormatter, che indica che vogliamo salvare i nostri dati in formato binario.
Successivamente viene instanziato un oggetto dat System.IO.Stream Writer che apre un canale di comunicazione in scrittura con il file person.bin come indicato nel parametro passato al metodo.
Infine grazie al metodo serialize dell'oggetto bin, che prende come parametro lo stream su cui salvare i dati e il nome dell'oggetto che deve essere salvato, trascriviamo lo stato dell'oggetto sull'hard disk.
Al successivo riavvio dell'applicazione viene generato l'evento Load della form. Al suo interno vengono di nuovo instanziati il formatter e lo stream, e, questa volta, grazie al metodo deserialize che recupera lo stato dell'oggetto precedentemente salvato, tutto viene riportato allo stato originario.

REALIZZIAMOLO IN VB.NET

Fino al punto 4 i passi sono identici. Il codice che implementa la classe person in Visual Basic è il seguente:

```
<Serializable()> _
Public Class person
    Private p_nome As String
    Private p_cognome As String
    Private p_organizzazione As String

    Public Sub person()
    End Sub

    Property Nome() As String
        Get
            Return p_nome
        End Get
        Set(ByVal Value As String)
            p_nome = Value
        End Set
    End Property
Property Cognome() As String
        Get
            Return p_cognome
        End Get
        Set(ByVal Value As String)
```

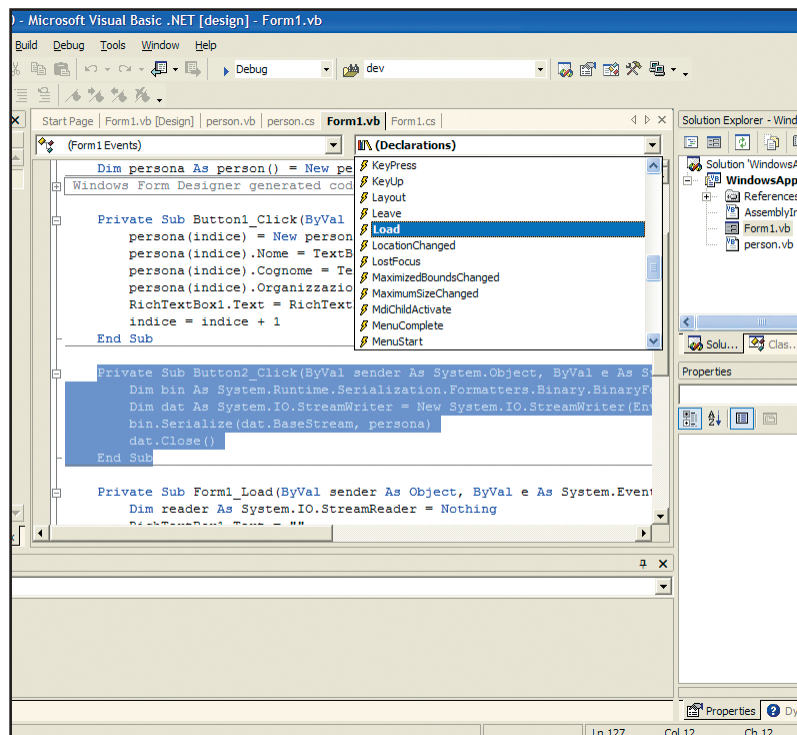
```
            p_cognome = Value
        End Set
    End Property

    Property Organizzazione() As String
        Get
            Return p_organizzazione
        End Get
        Set(ByVal Value As String)
            p_organizzazione = Value
        End Set
    End Property

End Class
```

1 Torniamo alla finestra di disegno della form e clicchiamo due volte sul bottone “Add” per creare il template di gestione dell’evento “OnClick”.
Il nostro codice sarà il seguente:

```
Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                            Handles Button1.Click
        persona(indice) = New person
        persona(indice).Nome = TextBox1.Text
        persona(indice).Cognome = TextBox2.Text
        persona(indice).Organizzazione =
                                    TextBox3.Text
        RichTextBox1.Text = RichTextBox1.Text +
                    persona(indice).Nome + " "
                    + persona(indice).Cognome + "
                    " + persona(indice).Organizzazione
        indice = indice + 1

    End Sub
```

2 Sempre nell’interfaccia di disegno della form cliccando due volte sul pulsante “Save” e inseriamo il codice di gestione dell’evento

```
Private Sub Button2_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                            Handles Button2.Click
        Dim bin As System.Runtime.Serialization
            .Formatters.Binary.BinaryFormatter = New
                System.Runtime.Serialization
                .Formatters.Binary.BinaryFormatter
        Dim dat As System.IO.StreamWriter = New
                        System.IO.StreamWriter(
        Environment.CurrentDirectory + "\persona.bin")
        bin.Serialize(dat.BaseStream, persona)
        dat.Close()

    End Sub
```

3 Dal solution explorer clicchiamo su Form1 con il tasto destro del mouse e dall’elenco a discesa selezioniamo view code. Dalla casellina a sinistra scegliamo “Form1 Events” e dagli eventi scegliamo “Load”

4 Il codice da inserire per la gestione dell’evento Load della form è il seguente:

```
Private Sub Form1_Load(ByVal sender As Object,
                    ByVal e As System.EventArgs)
                            Handles MyBase.Load
        Dim reader As System.IO.StreamReader =
                                        Nothing
        RichTextBox1.Text = ""
        Try
            Dim bin As System.Runtime.Serialization
                .Formatters.Binary.BinaryFormatter =
                    New System.Runtime .Serialization
                    .Formatters.Binary.BinaryFormatter
            reader = New System.IO.StreamReader(
                        Environment.CurrentDirectory
                                + "\persona.bin")
                persona = bin.Deserialize(
                            reader.BaseStream)
            reader.Close()
            For Each dato As person In persona
                RichTextBox1.Text += dato.Nome + " "
                        + dato.Cognome + Chr(13)
            Next
        Catch
        Finally
            If (Not reader Is Nothing) Then
                reader.Close()
            End If
        End Try
```

CONTARE LE LINEE DI UN FILE CON LE REGULAR EXPRESSIONS

SPESSO NEI NOSTRI PROGRAMMI INTERESSA CONOSCERE IL NUMERO DI LINEE CHE COSTITUISCONO UN FILE DI TESTO, UNO DEI METODI POSSIBILI È QUELLO DI LEGGERE TUTTE LE RIGHE ED INCREMENTARE UN CONTATORE. UN'ALTRA POSSIBILITÀ È DATA DALLE ESPRESSIONI REGOLARI, DATO CHE OGNI RIGA È TERMINATA DA UN CARATTERE '\N' BASTA VERIFICARE LA LORO OCCORRENZA ALL'INTERNO DEL FILE.

C#

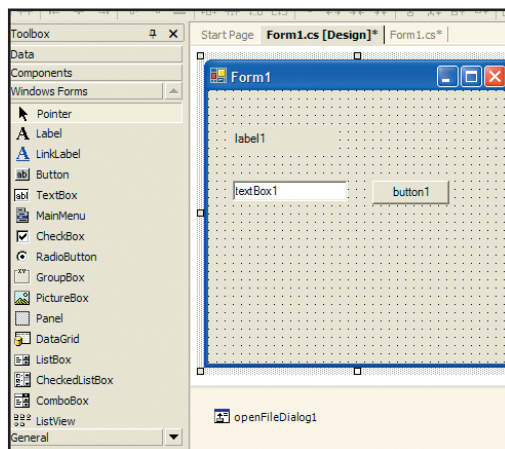
VISUAL BASIC.NET

I NAMESPACE DA IMPORTARE

Per Visual Basic è necessario aggiungere i seguenti import all'inizio del codice che rappresenta la classe:

```
Imports System.IO
Imports System.Text
    .RegularExpressions
```

FACCIAMOLO IN C#

1 Posizioniamoci sulla Windows Form C# principale del programma e posizioniamo su di essa una label, una TextBox, un bottone e un componente openfiledialog

2 Clicchiamo due volte sul bottone per generare il template di gestione dell'evento "click".
Il codice da inserire è il seguente:

```
private void button1_Click(object sender,
                          System.EventArgs e)
{
    openFileDialog1.ShowDialog();
    textBox1.Text = openFileDialog1.FileName;
    label1.Text = getLines(openFileDialog1.FileName);
}
```

3 Non ci resta che definire il codice per il metodo GetLines. Possiamo inserire questo codice poco prima di quello che gestisce l'evento click:

```
private static string getLines(string filename)
{
        string file=filename;
        StreamReader r=File.OpenText(file);
        string content=r.ReadToEnd();
        r.Close();
        Regex rex=new Regex("\n");
        int count=rex.Matches(content).Count;
        return count.ToString();
}
```

COSA SONO LE REGULAR EXPRESSION?

Si indica con questo termine o anche con l'abbreviativo Regex, un metalinguaggio utilizzato per costruire interrogazioni su file di testo. Ad esempio la regex "ioProgrammo" indica che si intendono ricercare all'interno del testo tutte le occorrenze della parola "ioProgrammo". La regex "io?rogrammo" indica che si vogliono trovare tutte le occorrenze di una parola che ha come prime due lettere "io" , segue una lettera qualunque, segue la stringa "rogammo". La regex "io*" indica che si vogliono trovare tutte le parole che iniziano per "io". Gli esempi che abbiamo proposto fin qui, sono molto semplici, tuttavia con le regex è possibile effettuare combinando i vari elementi della sintassi, ricerche veramente complesse.

COME FUNZIONA

In relazione all'evento click viene mostrata una finestra di dialogo che consente di selezionare il file da analizzare. Il valore della TextBox1 viene cambiato in relazione al file selezionato dall'utente.
Il nome del file viene passato come parametro al metodo GetLines, che restituisce una stringa e valorizza il contenuto della label.
Il metodo GetLines apre fisicamente un canale di lettura per il file, lo legge e inserisce il suo contenuto nella variabile content. A questo punto viene fatto uso delle Regex che confrontano il contenuto di content con il valore "\n" che indica il carattere di fine linea, e ne restituiscono il numero di occorrenze.

REALIZZIAMOLO IN VB.NET

Fino al punto 2 i passi sono identici. Il codice a inserire per la gestione dell'evento Click è il seguente

```
Private Sub Button1_Click(ByVal sender As
                         System.Object, ByVal e
               As System.EventArgs) Handles Button1.Click
    OpenFileDialog1.ShowDialog()
    TextBox1.Text = OpenFileDialog1.FileName
    Label1.Text = getLines(
                         OpenFileDialog1.FileName)
  End Sub
```

Il metodo GetLines diventa invece:

```
Private Function getLines(ByVal filename As
                                String) As String
    Dim file As String = filename
    Dim r As StreamReader =
                    System.IO.File.OpenText(file)
    Dim content As String = r.ReadToEnd
    r.Close()
    Dim rex As Regex = New Regex("\n")
    Dim count As Int16 = rex.Matches(
                                content).Count
        Return count.ToString()
    End Function
```

I NAMESPACE DA IMPORTARE

Perché tutto funzioni è necessario aggiungere il seguente codice, all'inizio del programma nella sezione dedicata ai namespace:

```
using System.IO;
using System.Text.Regular Expressions;
```

CHE COS'È UN THREAD?

CON UNA BUONA DOSE DI APPROSSIMAZIONE, SI PUÒ DIRE CHE IN UN PROGRAMMA LE OPERAZIONI VENGONO ESEGUITE IN SEQUENZA, E CHE IN OGNI DATO MOMENTO PUÒ ESSERE ESEGUITA UNA SOLA OPERAZION. I THREAD CONSENTONO DI ESEGUIRE OPERAZIONI IN PARALLELO

C#

Ad esempio in uno spezzone di codice composto da tre righe, verrà eseguita prima la prima riga, poi la seconda e poi la terza. La seconda e di conseguenza la terza non saranno eseguite fino a che la prima non sarà terminata.
Questo modo di operare può rappresentare in qualche occasione un problema. In questo stesso numero di ioProgrammo abbiamo proposto un esempio di "ping" di un sito web. Vi accorgerete che eseguendolo, l'interfaccia rimane bloccata fino a che non viene restituito un esito.
I thread viceversa sono spezzoni di codice asincrono che non vengono eseguiti in sequenza ma in parallelo. Questo introduce delle complicazioni nella gestione del codice, ma risolve i problemi di cui sopra.
L'esempio del Ping, modificato utilizzando i thread è il seguente.

FACCIAMOLO IN C#

1 Importiamo i namespace che serviranno per fare gestire i thread. Possiamo aggiungerli in alto nella sezione che contiene i vari namespace:

```
using System.Threading;
using System.Runtime.Remoting.Messaging;
```

2 dichiariamo un *delegate*

```
delegate bool UrlPing(int retries,string url);
```

3 Clicchiamo due volte sul bottone per generare il template del codice che dovrà gestire il click. Il codice relativo sarà:

```
private void button1_Click(
                    object sender, System.EventArgs e)
{
    UrlPing u = new UrlPing(pingUrl);
    u.BeginInvoke(10,textBox1.Text,new
                    AsyncCallback(AfterPing),"status");
}
```

4 Nessuna variazione verrà fatta al metodo *pingUrl*, che rimane:

```
private bool pingUrl(int retries, string URL)
{  bool connected = false;
   TcpClient client = new TcpClient();
   try
   {       client.Connect(URL, 80);
           client.Close();
           connected = true; }
   catch (SocketException)
   { connected = false; }
   return connected;
}
```

5 Scriviamo il codice di gestione della funzione di CallBack *AfterPing:*

```
public void AfterPing (IAsyncResult result)
  {  AsyncResult async = (AsyncResult) result;
     UrlPing fetcher = (UrlPing) async.AsyncDelegate;
     if (fetcher.EndInvoke(result) == true)
     {  this.label1.Text="Ok";
     }
     else
     {  this.label1.Text="Ko";
```

```
        }
    };
}
```

COME FUNZIONA

Alla pressione del bottone, viene richiamato il codice di gestione dell'evento Click. All'interno di questo codice, tramite la dichiarazione

```
UrlPing u = new UrlPing(pingUrl);
```

instanziamo un oggetto u di classe UrlPing. UrlPing è un delegate, incorpora cioè un metodo con una specifica signature. Siamo costretti ad usare questo stratagemma, perchè non possiamo passare dei parametri a un metodo che deve funzionare in un thread. In questo modo possiamo invece utilizzare il seguente spezzone di codice:

```
u.BeginInvoke(10,textBox1.Text,new
                    AsyncCallback(AfterPing),"status");
```

questo ci consente di creare un nuovo thread per il metodo pingUrl, inoltre possiamo passare i parametri relativi al metodo e definire una funzione di callback che verrà eseguita quando il thread sarà completato. Al completamento del thread viene richiamato il metodo AfterPing che restituisce l'esito in un oggetto di tipo AsyncResult

COME POSSO APRIRE UNA PAGINA HTML NEL BROWSER PREDEFINITO?

Per aprire un file html nel browser di sistema predefinito basta creare una istanza della classe *System.Diagnostics .Process* utilizzando come parametro il percorso completo del file o l'indirizzo http.

```
System.Diagnostics.Process.Start(
                                        "file.htm");
```

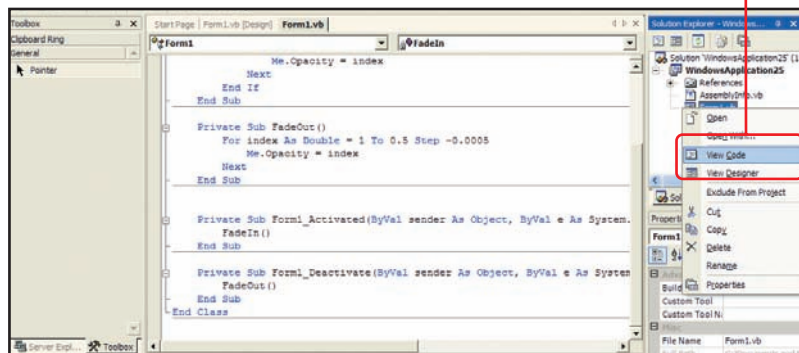
COSTRUIRE UN FORM CON EFFETTO "FADE"

PER CREARE UN FORM CHE COMPARE LENTAMENTE ALL'AVVIO E SCOMPARE ALTRETTANTO LENTAMENTE ALLA CHIUSURA SI PUÒ UTILIZZARE LA PROPRIETÀ "OPACITY" DELLE FORM, CHE È RESPONSABILE DELLA LORO TRASPARENZA

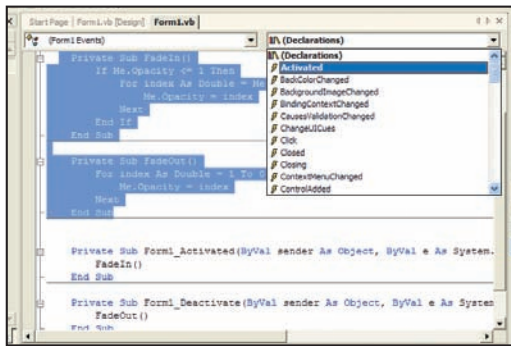
VISUAL BASIC.NET

C#

REALIZZIAMOLO IN VB.NET

1 Clicchiamo con il tasto destro nel Solution Explorer, alla voce che rappresenta la nostra form e dal menu a tendina selezioniamo *"view code"*

2 Aggiungiamo le due funzioni che ci serviranno per gestire l'effetto fade in entrata e in uscita:

```
Private Sub FadeIn()
    If Me.Opacity <= 1 Then
        For index As Double = Me.Opacity To 1
                                        Step 0.0005
            Me.Opacity = index
        Next
    End If
End Sub
Private Sub FadeOut()
    For index As Double = 1 To 0.5 Step -0.0005
        Me.Opacity = index
    Next

End Sub
```

3 Nella combo box degli oggetti, a sinistra, clicchiamo su "Form1 Events" e nella combo box a destra selezioniamo "Activated" questo ci consentirà di creare il template del codice che sarà eseguito quando un utente selezionerà la form:

4 Nel template appena generato inseriamo il codice di gestione

```
Private Sub Form1_Activated(ByVal sender As
        Object, ByVal e As System.EventArgs)
        Handles MyBase.Activated
    FadeIn()
End Sub
```

5 Ripetiamo il passo 3, avendo cura questa volta di selezionare l'evento "deactivate". Il codice da associare all'evento è

```
Private Sub Form1_Deactivate(ByVal sender As
        Object, ByVal e As System.EventArgs)
        Handles MyBase.Deactivate
    FadeOut()
End Sub
```

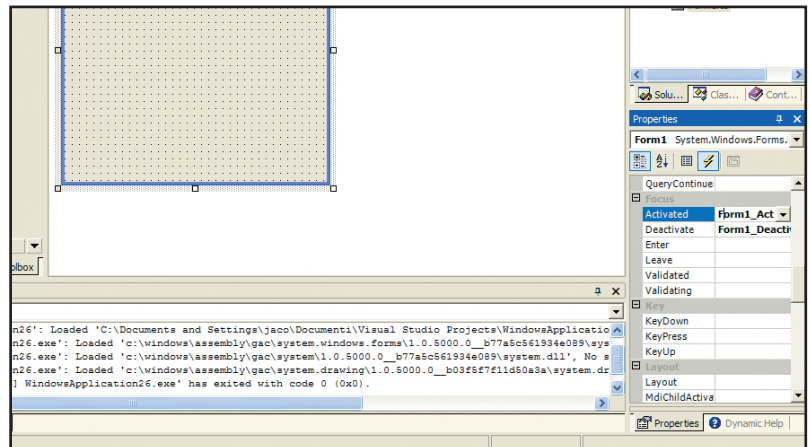
COME FUNZIONA

Quando la finestra non ha più il focus, si scatena un evento di tipo deactivate. All'interno di questo evento viene richiamata la funziona fadeOut che tramite un ciclo di for decrementa il valore della proprietà opacity che controlla la trasparenza della form. Allo stesso modo quando l'utente restituisce il focus alla finestra viene richiamata la funzione fadeIn che incrementa di nuovo la proprietà opacity fino al valore 1, che è anche il suo massimo valore e corrisponde ad una form non trasparente, viceversa il valore 0 fa si che la form sia completamente trasparente.

FACCIAMOLO IN C#

Fino al punto 2 i passi sono identici. I metodi FadeIn e FadeOut diventano:

```
private void FadeOut()
{ for (double index=1; index > 0.5;index-=0.0005)
   this.Opacity = index;
}
private void FadeIn()
{ if (this.Opacity <= 1)
   for (double index=0.5; index <= 1;index+=0.0005)
   this.Opacity = index;
}
```

1 Per creare automaticamente il template in relazione agli eventi Activate e Deactivate è necessario portarsi nella tabella degli eventi e cliccare due volte sull'evento relativo

2 Il codice di gestione degli eventi activate e deactivate sarà:

```
private void Form1_Activated(
        object sender, System.EventArgs e)
{  FadeIn();}
private void Form1_Deactivate(
        object sender, System.EventArgs e)
{
   FadeOut();
}
```

COSA SONO LE ENUMERAZIONI?

In .NET le enumerazioni sono delle istanze della classe System.Enum, che deriva dalla classe ValueType. La classe Enum fornisce diversi metodi per lavorare con le enumerazioni, fra i quali i metodi statici GetNames che permette di ottenere i membri dell'enumerazione, e Parse che converte una stringa in un oggetto del tipo enumerativo:

```
enum GiorniSettimana {
      Lun, Mar, Mer, Gio, Ven, Sab, Dom};
...
Type giorni = typeof(GiorniSettimana);
Console.WriteLine("I giorni della
   settimana ed i rispettivi valori sono:");
foreach ( string giorno in Enum.GetNames(
        giorni) )
 Console.WriteLine( "{0,-5}= {1}",
        giorno, Enum.Format( giorni,
   Enum.Parse(giorni, giorno), "d") );
```

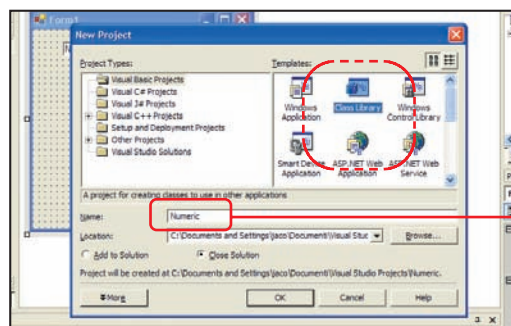
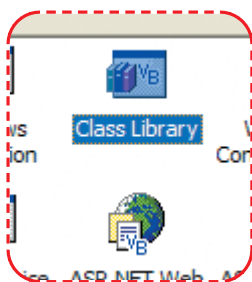
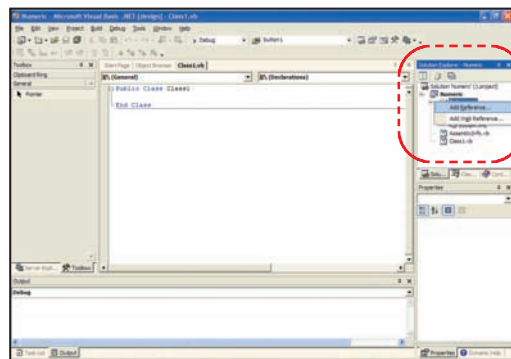
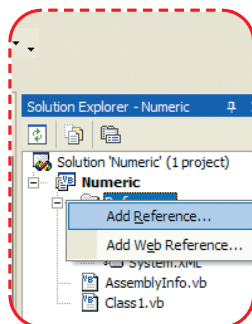
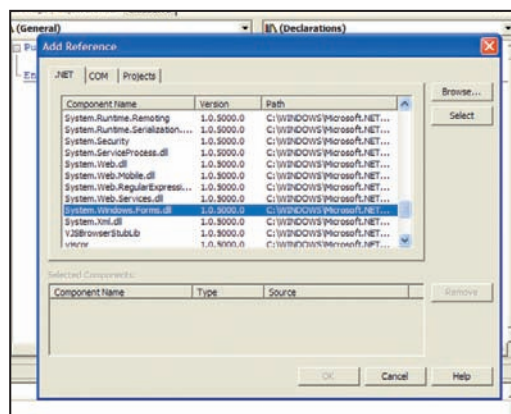
COME CREARE UN CONTROLLO CHE ACCETTA SOLO NUMERI?

TALVOLTA È UTILE CREARE TEXTBOX CHE PERMETTONO L'INSERIMENTO DI SOLI CARATTERI NUMERICI. IMPARIAMO COME CREARE UN SEMPLICE QUANTO UTILE CONTROLLO RIUTILIZZABILE

VISUAL BASIC.NET

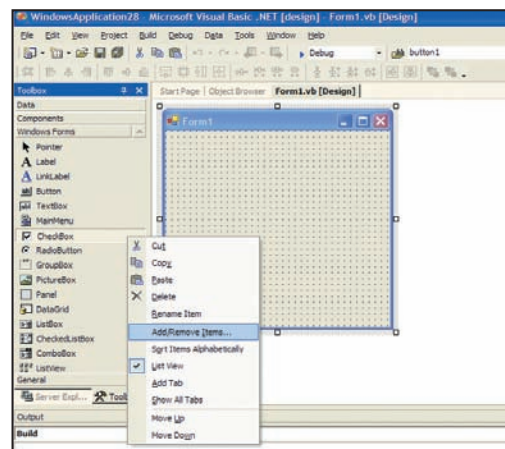
C#

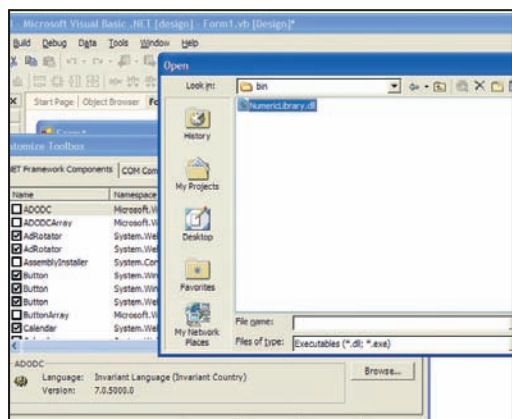
FACCIAMOLO IN VB.NET

1 Iniziamo da "File/New Project" e scegliamo "class library". Scegliamo un nome significativo per la nostra libreria, ad esempio *"Numeric"*, clicchiamo su Ok e andiamo avanti.

2 Nel solution explorer clicchiamo su references e di seguito su "add reference"

3 Dalla successiva schermata selezioniamo System.windows.forms.dll, cliccando per due volte sulla riga corrispondente, e subito dopo su ok

4 Il codice di gestione della libreria sarà:

```
Imports System.Windows.Forms

Public Class NumericTextBox
    Inherits System.Windows.Forms.TextBox

    Private Sub NumericTextBox_KeyPress(ByVal
                    sender As Object, ByVal e
        As System.Windows.Forms.KeyPressEventArgs)
                    Handles MyBase.KeyPress
        If ((Not Char.IsDigit(e.KeyChar)) And (Not
                Char.IsControl(e.KeyChar))) Then
            e.Handled = True
        End If
    End Sub

End Class
```

5 Compiliamo la libreria usando la combinazione di tast CTRL+SHIFT+B. Chiudiamo il tutto e iniziamo una nuova soluzione di tipo Windows Application

6 Per importare il nostro nuovo controllo nella toolbox, clicchiamo con il tasto destro sulla barra degli strumenti e scegliamo "Add/Remove Items"

7 Nella schermata successiva clicchiamo sul tasto browse e di seguito andiamo alla ricerca della libreria precedentemente compilata nell'hard disk. Una volta completato il processo troverete il nuovo controllo nella toolbox e

potrete utilizzarlo in modo classico trascinandolo nella form. Noterete che non sarà possibile inserire nella casella di testo, caratteri diversi da quelli numerici

COME FUNZIONA

Il nuovo controllo eredita da System.Windows .Forms.TextBox. Viene poi creato un gestore personalizzato per l'evento KeyPress. È sufficiente poi aggiungere il nuovo controllo alla toolbox per poterlo utilizzare esattamente come un controllo nativo.

FACCIAMOLO IN C#

Fino al punto 4 i passi sono uguali eccetto che per la scelta del progetto, che in questo caso deve essere in C#. Il codice di gestione della libreria diventa:

```
using System.Windows.Forms ;
namespace ClassLibrary2
{ /// <summary>
  /// Summary description for Class1.
  /// </summary>
  public class NumericText:
                              System.Windows.Forms.TextBox
  { public NumericText()
    {  this.KeyPress += new KeyPressEventHandler(
                                   NumericTextBox_KeyPress);
      //
      // TODO: Add constructor logic here
      //
    }
    private void NumericTextBox_KeyPress(object
                              sender,System.Windows.Forms
                                     .KeyPressEventArgs e)
    {  if (!(char.IsDigit(e.KeyChar))
                               & !(char.IsControl(e.KeyChar)))
        {  e.Handled=true; }
}
```

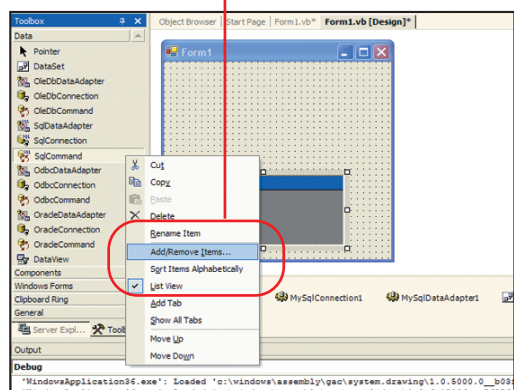
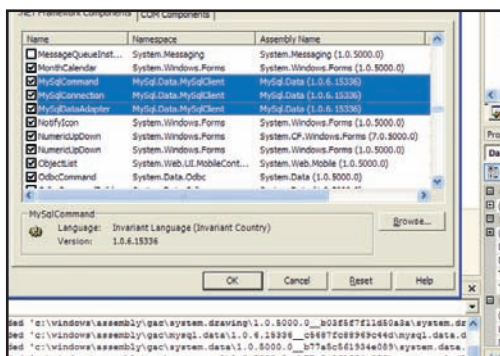
COME POSSO USARE UNA DATAGRID CON MYSQL?

UN METODO UTILE PER INTERFACCIARE MYSQL CON APPLICAZIONI .NET

VISUAL BASIC.NET

C#

FACCIAMOLO IN VB.NET

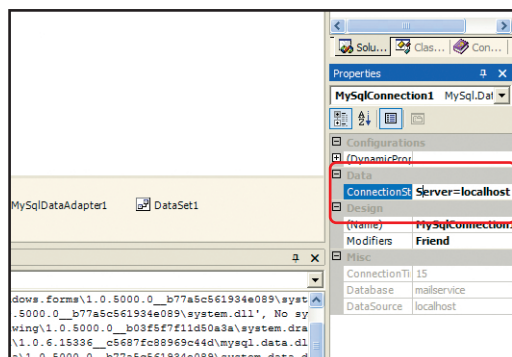
1 Iniziamo aggiungendo alla toolbox i controlli necessari alla gestione di MySQL, per farlo, clicchiamo con il tasto destro del mouse sulla toolbox e dal menu a tendina che compare selezioniamo *"Add/Remove item"*

2 Nella finestra che segue selezioniamo "MySQL Command", "MySQL Connection", "MySQL Data Adapter". Inoltre aggiungiamo nei references la libreria relativa a csharpziplib

COME INIZIARE

Prima di tutto c'è bisogno di installare il .NET/Provider, lo trovate in allegato al CD di ioProgrammo, oppure all'indirizzo: http://dev.mysql.com /downloads /connector/net/1.0.html.

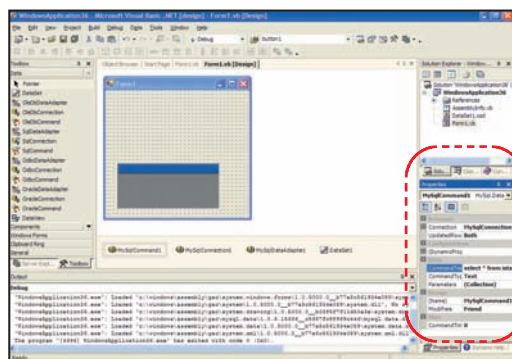
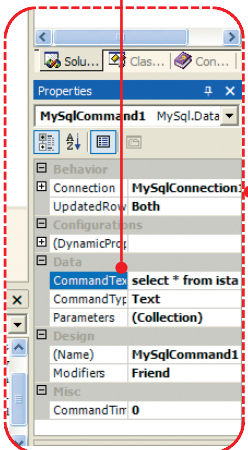
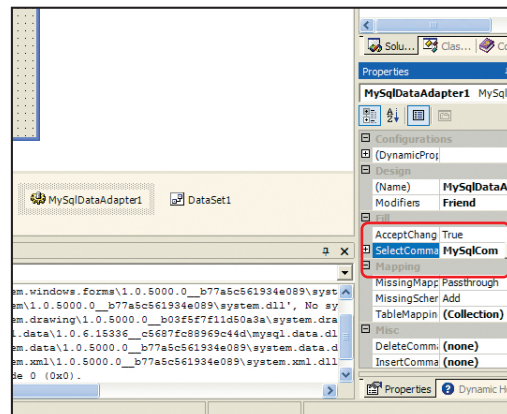
È buona norma installare anche le SharpZip Library, le trovate nel CD allegato alla rivista, oppure sul sito http://www.icsharpcode .net /OpenSource/SharpZipLib/Default.aspx.

3 Dalla toolbox trasciniamo sulla form un "MySQL Command", un "MySQL Data Adapter", una "MySQL Connection" un "DataSet" e infine una "DataGrid"

4 Nella property "Connection String" relativa al componente *"MySQL Connection"* inseriamo la stringa con i parametri di connessione. Nel nostro caso

```
Server=localhost;Database=test;Uid=
                    ioProgrammo;Pwd=pluto
```

5 Nella property "CommandText" relativa al componente *"MySQL Command"*, inseriamo la query di interrogazione al database. Nel nostro caso

```
select * from istat
```

Assicuriamoci anche che nella property Connection dello stesso componente sia selezionato *"MySQLConnection1"* ovvero il riferimento al componente inizializzato nel passo 4

6 Selezioniamo il DataAdapter e assicuriamoci che nella proprietà *"SelectCommand"* sia selezionato il riferimento al componente "MySQL command" instanziato nel passo 5. Nel nostro caso è "MySQLCommand1"

7 Passiamo alla modalità di visualizzazione del codice, cliccando nel solution explorer su "view code", e interveniamo sull'evento "Load". Il codice da inserire è il seguente:

```
Private Sub Form1_Load(ByVal sender As Object,
                        ByVal e As System.EventArgs)
                                Handles MyBase.Load
    MySqlDataAdapter1.Fill(DataSet1, "Istat")
    DataGrid1.DataSource() =
                            DataSet1.Tables("Istat")
End Sub
```

Dove "Istat" in questo caso è il nome della tabella con cui abbiamo riempito la query

INSTALLARE LE SHARPZIPLIB

All'interno dello zip, trovate una serie di file .bat, lanciate una console dal prompt dei comandi, portatevi nella directory dove avrete decompresso lo zip, e digitate installgac.bat. Se tutto va a buon fine otterrete il messaggio di corretta installazione, viceversa sarà necessario aggiungere il percorso relativo al comando GacUtils nella variabile d'ambiente "Path".

COME FUNZIONA

Il componente MySQL Connection si occupa di stabilire la connessione con MySQL. Il componente MySQL Command esegue la query sul database. Il dataAdapter funge come ponte tra il provider .NET e la rappresentazione dei dati in memoria, che è mantenuta nella struttura di dataset.
L'intero dataset viene poi rappresentato in una griglia.

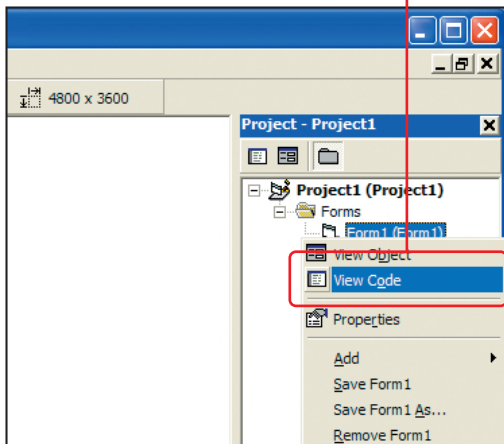
FACCIAMOLO IN C#

I passi dall'1 al 6 rimangono invariati, il codice che invece riempie la datagrid è il seguente

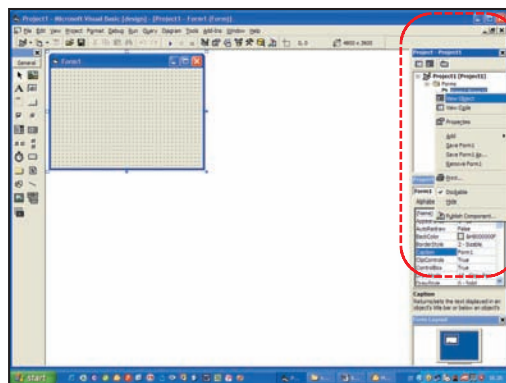
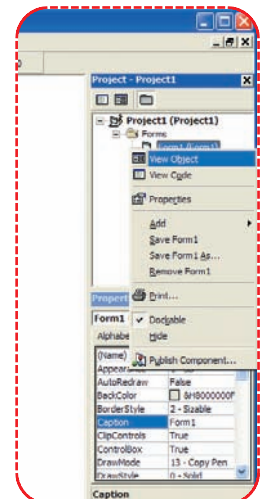
```
private void Form1_Load(object sender,
                              System.EventArgs e)
{
  mySqlDataAdapter1.Fill(dataSet1,"Istat");
  dataGrid1.DataSource = dataSet1.Tables["Istat"];
}
```

CREARE UNA FINESTRA DI FORMA CIRCOLARE IN VISUAL BASIC

PER COSTRUIRE UNA FINESTRA DI FORMA CIRCOLARE DOBBIAMO USARE ALCUNE FUNZIONI API. IN PARTICOLARE NELLA PARTE DICHIARATIVA DELLA FORM BISOGNA INSERIRE LA DEFINIZIONE DELLE SEGUENTI FUNZIONI: CREATEELLIPTICRGN, CREATERECTRGN, COMBINERGN, SETWINDOWRGN.

1 Portatevi nel Project Explorer e con il tasto destro del mouse selezionate la form. Dal menu che segue selezionate *View Code*

Nella finestra che compare aggiungete il seguente codice

```
Private Declare Function CreateEllipticRgn Lib _
"gdi32" (ByVal x1 As Long, ByVal y1 As Long, _
ByVal X2 As Long, ByVal Y2 As Long) As Long

Private Declare Function CreateRectRgn Lib _
"gdi32" (ByVal x1 As Long, ByVal y1 As Long, _
ByVal X2 As Long, ByVal Y2 As Long) As Long

Private Declare Function CombineRgn Lib _
"gdi32" (ByVal hDestRgn As Long, ByVal hSrcRgn1 As _
Long, ByVal hSrcRgn2 As Long,
                ByVal nCombineMode As Long) As Long

Private Declare Function SetWindowRgn Lib _
"user32" (ByVal hWnd As Long, ByVal hRgn As _
              Long, ByVal bRedraw As Boolean) As Long

Dim iden As Long
Const RGN_XOR = 3
Const RGN_AND = 1

Private Function Crearegione() As Long
 Dim RegCombinate As Long
 Dim NuovaRegione As Long
 Dim regForm As Long
 regForm = CreateRectRgn(0, 0, Width, Height)
 RegCombinate = CreateRectRgn(0, 0, 0, 0)
 iden = CombineRgn(RegCombinate, RegCombinate,
                              regForm, RGN_XOR)
 NuovaRegione = CreateEllipticRgn(
                                74, 108, 464, 498)
 iden = CombineRgn(RegCombinate, RegCombinate,
                          NuovaRegione, RGN_AND)
 Crearegione = RegCombinate

End Function
```

2 Ritornate alla form cliccando con il tasto destro del mouse nel project explorer e nel menu che segue in “view object”

3 Sulla form trascinate un bottone dalla toolbox e ciccatevi due volte sopra per generare il template del codice che dovrà gestire l’evento onclick, il codice è il seguente:

```
Private Sub circolare_Click()
 Me.Width = 8175
 Me.Height = 8100
 iden = SetWindowRgn(hWnd, Crearegione, True)
End Sub
```

COME FUNZIONA

La Crearegione prima combina una regione che copre completamente la form con una regione vuota poi interseca il risultato con una regione circolare. I valori dei parametri della CreateEllipticRgn e delle dimensioni della form (cioè i valori delle proprietà Width e Height) sono stati valutati in modo che l’ellisse sia una circonferenza che ricopra la Form. Da notare come abbiamo usato le API di sistema in modo conveniente in Visual Basic.

COME CATTURARE LO SCREENSHOT DEL DESKTOP

DI SEGUITO VIENE MOSTRATA UNA SEMPLICE CLASSE JAVA PER LA CATTURA DELL'IMMAGINE VISUALIZZATA SULLO SCHERMO, AD ECCEZIONE DEL PUNTATORE DEL MOUSE

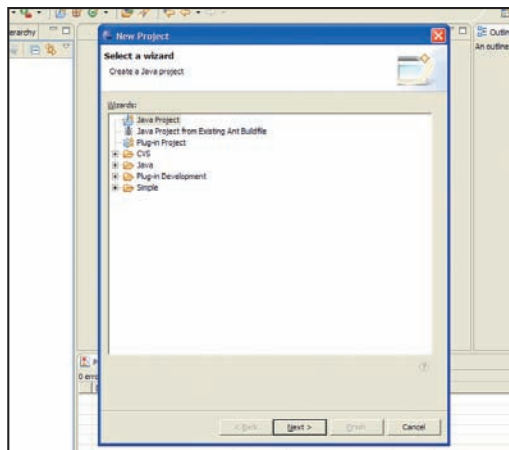
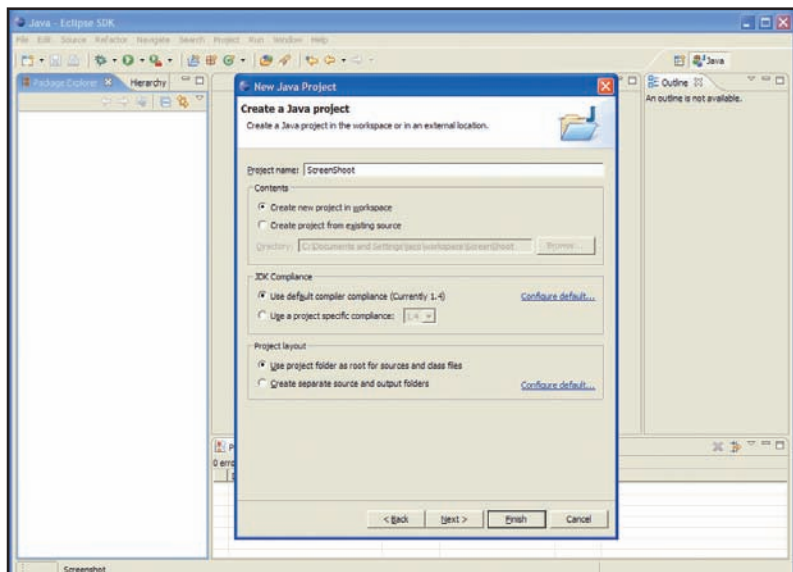
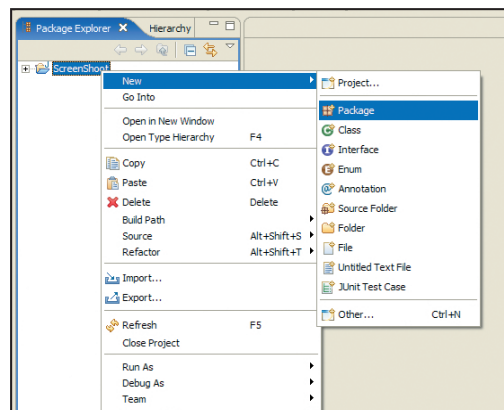
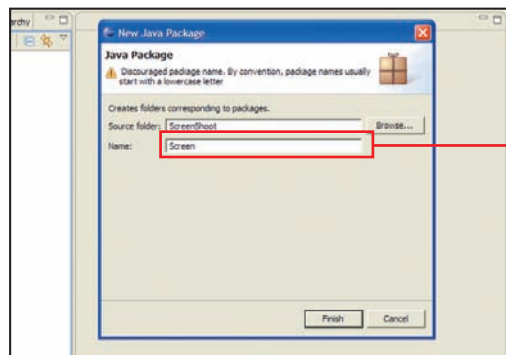
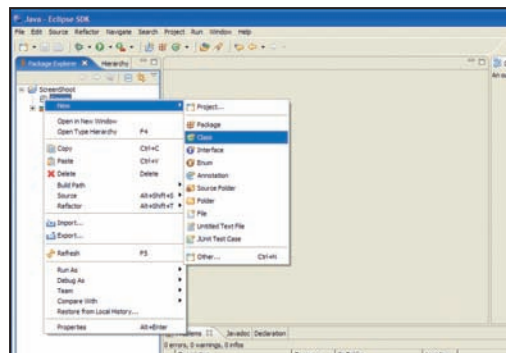
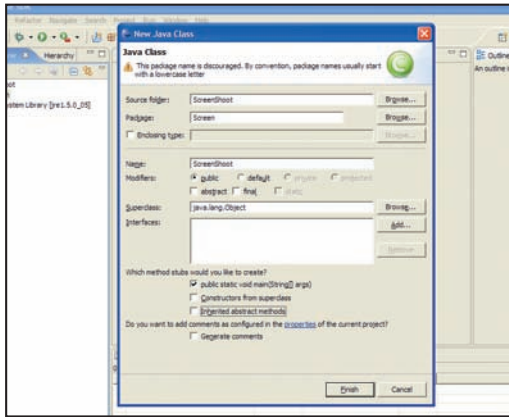
JAVA

CD WEB

Eclipse 3.1, J2SE 5.0

cdrom.ioprogrammo.it

FACCIAMOLO IN JAVA

1 Iniziamo da *"File/New Project"*. Scegliamo *"Java Project"* e clicchiamo su *Next*.

2 Nella schermata successiva inseriamo un nome significativo per il progetto ad esempio *"ScreenShot"* e clicchiamo su *"Next"*.
Nelle schermate successive clicchiamo semplicemente su *Next* fino a quando possibile, e infine su *"finish"*

3 Creiamo un nuovo package cliccando con il tasto destro del mouse sul nome del progetto a sinistra nel *"Package Explorer"* e selezionando *New/Package*.

4 Diamo un nome significativo al pacchetto, ad esempio *"Screen"* e proseguiamo

5 Clicchiamo con il tasto destro del mouse sul package appena creato e selezioniamo il menu *"New/Class"*

6 Nella schermata che segue preoccupiamoci di dare un nome significativo alla classe, ad esempio *"ScreenShoot"*. Controlliamo anche che sia selezionato il flag *"Public static void main (string[] args)*. Quando siamo pronti clicchiamo su *finish*

7 Nel template del codice che viene generato aggiungiamo prima di tutto gli *import* corretti che ci servono per il nostro progetto

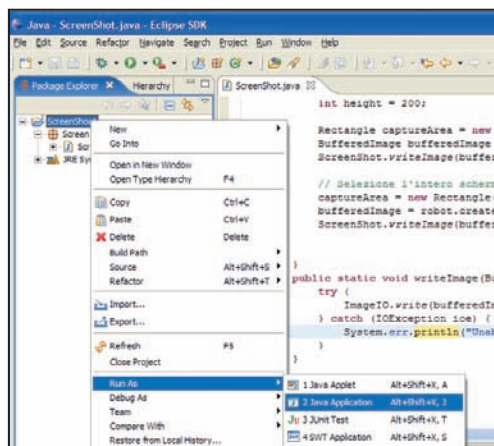
```
package Screen;
import java.awt.AWTException;
import java.awt.Rectangle;
import java.awt.Robot;
import java.awt.Toolkit;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
public class ScreenShoot
{
```

8 Definiamo il *Main* dell'applicazione come segue

```
public static void main(String[] args)
  {
    Robot robot = null;
    try {
          robot = new Robot();
    }
    catch (AWTException e)
    {
          System.err.println("Unable to instantiate
              Robot. Caused by: " + e.getMessage());
          System.exit(-1);
      }
      // Selezione una particolare zona dello schermo
      int x = 0;
      int y = 0;
      int width = 200;
      int height = 200;
      Rectangle captureArea = new Rectangle(
                                  x, y, width, height);
      BufferedImage bufferedImage =
              robot.createScreenCapture(captureArea);
      ScreenShot.writeImage(bufferedImage,
                                      "screenshot1");
      // Selezione l'intero schermo
      captureArea = new Rectangle(
              Toolkit.getDefaultToolkit().getScreenSize());
          bufferedImage = robot.createScreenCapture(
                                          captureArea);
          ScreenShot.writeImage(
                      bufferedImage, "screenshot2");
      }
```

9 Infine definiamo la *WriteImage* che il metodo che viene richiamato per scrivere fisicamente le immagini sull'Hard Disk

```
public static void writeImage(BufferedImage
                      bufferedImage, String fileName)
  {
      try {
          ImageIO.write(bufferedImage, "png", new
                          File(fileName + ".png"));
      }
      catch (IOException ioe)
      {
          System.err.println("Unable to write image :
                          " + ioe.getMessage());
      }
```

10 Lanciamo il tutto cliccando con il tasto destro del mouse sul nome del progetto e selezionando *Run/As Application*. Nella schermata successiva digitiamo *Screenshot* e poi clicchiamo su *Invio*.

COME FUNZIONA

Il cuore dell'applicazione è l'oggetto robot di Classe Robot. Normalmente questa classe viene utilizzata per creare e gestire gli eventi in modo particolare quando si vogliono eseguire test sull'applicazione o demo. Fra gli altri un oggetto di classe Robot può usare il metodo CreateScreenCapture che cattura l'immagine dallo schermo e la salva in un oggetto di tipo BufferedImage, esattamente come abbiamo fatto noi. Il metodo writeImage non fa altro che usare la classe ImageIO e il suo metodo write per scrivere l'immagine sull'hard disk.

COME LEGGERE UN FILE XML

XML È ORMAI UNO STANDARD PER LA COMUNICAZIONE DEI DATI E PER MOLTO ALTRO ANCORA. VEDIAMO COME USARLO CON I PRINCIPALI LINGUAGGI

JAVA

C#

VISUAL BASIC.NET

PHP

La flessibilità di XML consiste nel poter utilizzare file composti ad esempio come segue:

```
<?xml version="1.0" encoding="UTF-8"?>
<item>dato</item>
```

qualunque programma in grado di leggere il file XML in questione sarà in grado di interpretare quanto in esso contenuto. Proprio perché le regole di XML sono standard e certe non bisogna reinventare i parser ogni volta.

FACCIAMOLO CON JAVA

1 Definiamo una nuova classe xmldom, composta come segue:

```
package reader;

import java.io.*;
import javax.xml.parsers.*;
import org.w3c.dom.Document;
import org.xml.sax.SAXException;

public class xmldom
{

  /**
  * @param args
  */
  String filename;
  Document document;

  public xmldom( String filename )
  {
     this.filename = filename;
  }

  public void parse() throws SAXException,
                                   IOException,
  ParserConfigurationException
  {

     DocumentBuilderFactory factory =
     DocumentBuilderFactory.newInstance();
     factory.setValidating(false);

     document = factory.newDocumentBuilder()
                        .parse(new File(filename));
  }

  public Document getDocument()
  {
     return document;
  }

}
```

I NAMESPACE DA IMPORTARE

È necessario importare il namespace XML inserendo la seguente riga all'inizio del codice

```
using System.Xml;
```

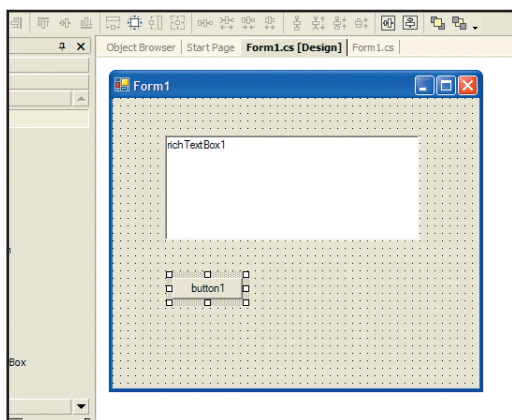
2 Successivamente una seconda classe farà uso dei metodi esposti nella prima. La seconda classe sarà così composta:

```
package reader;
import org.w3c.dom.*;
import javax.xml.*;

public class test
{

  /**
  * @param args
  */
  public static void main(String[] args)
  {

     // TODO Auto-generated method stub
     xmldom thisfile=new xmldom(
          "/home/fabio/workspace/javadom/reader
                                  /prova.xml");
     try
     {

        thisfile.parse();
        Node theNode = null;
        NodeList listaNodi = null;
        Document thisdocument =
                        thisfile.getDocument();
        listaNodi = thisdocument.getChildNodes();
        for( int i = 0; i < listaNodi.getLength(); i++ )
        {

           theNode = listaNodi.item(i);
           System.out.println( "Nome nodo: "+
             theNode.getFirstChild().getNodeValue());
        }

     }
     catch (Exception ex)
     {
        System.out.println("Error");
      }

   }

}
```

COME FUNZIONA

La classe xmldom espone due metodi, il metodo parse e il metodo getdocument. Nel costruttore accetta il nome di file xml da parserizzare. Il metodo parse si occupa di parserizzare il file, mentre il metodo getdocument ritorna un oggetto di tipo Document che può essere scansionato per trovare i dati che ci servono al suo interno.
Di fatto la seconda classe non fa altro che instanziare un oggetto xmldom, effettuare il parse e utilizzare un ciclo di for per scansionare tutti gli elementi all'interno del documento così costituito.

FACCIAMOLO IN C#

1 Per il nostro articolo ci avvarremo di un richtextbox e di un bottone. Trasciniamoli sulla form dalla toolbox

2 Clicchiamo due volte sul bottone per ottenere il template di gestione del codice. Il codice da inserire sarà il seguente:

```
private void button1_Click(object sender,
                            System.EventArgs e)
{

   XmlTextReader reader = new XmlTextReader(
              @"G:\098\javadom\reader\prova.xml");
   while (reader.Read())
   {

        if (!reader.IsEmptyElement)
        {
           richTextBox1.Text+=reader.Value;
        }
   }
}
```

FACCIAMOLO IN VISUAL BASIC.NET

I due passi iniziali rimangono identici, eccetto per il codice da inserire in reazione all'evento click, che diventa il seguente:

```
Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                                Handles Button1.Click
        Dim reader As XmlTextReader = New
                 XmlTextReader("G:\098\javadom
                                 \reader\prova.xml")
        While reader.Read()
            If Not reader.IsEmptyElement Then
                RichTextBox1.Text += reader.Value
            End If

        End While

    End Sub
```

FACCIAMOLO IN PHP

1 Creiamo un file *dato.xml* composto come segue:

```
<?xml version="1.0" encoding="UTF-8"?>
<item>dato</item>
```

2 Creiamo un file *index.php* contenente le seguenti istruzioni

```
<?
$dom = new DOMDocument();
$dom->load('file:///var/www/xml/dato.xml');
$dato= $dom->getElementsByTagName("item");
foreach ($dato as $node)
{
            print $node->firstChild->data."\n";
}
?>
```

ALTRI METODI DI LETTURA

Esistono diversi metodi per parserizzare un file XML in PHP. Un altro metodo utilizzabile è SAX, che è anche il metodo che è stato utilizzato con PHP4 e che comunque rimane disponibile anche con PHP5. Un altro metodo molto potente è costituito da DomXpath, in tal caso sarebbe possibile utilizzare il linguaggio di interrogazione Xquery per estrarre direttamente i dati che ci interessano.

COME FUNZIONA

Viene creato un nuovo oggetto di classe DOMDocument. Il contenuto del file XML viene caricato in memoria tramite il metodo Load.
Il file viene parserizzato e vengono recuperati tutti gli elementi il cui tag è <item>.
A questo punto abbiamo un'array di oggetti di classe $DOMNode. Il contenuto di un DomNode può essere recuperato con il metodo *firstchild->data* e stampato a schermo.

COME RECUPERARE L'ID DELL'ULTIMO RECORD INSERITO?

SPESSO È UTILE CONOSCERE L'ID DELL'ULTIMO RECORD INSERITO IN UNA TABELLA CHE ABBIA UNA CHIAVE AUTO_INCREMENT. MYSQL CI FORNISCE UNA FUNZIONE, LAST_INSERT_ID() CHE FA AL CASO NOSTRO. CHIAMANDOLA NELLA STESSA CONNESSIONE CHE HA FATTO L'INSERT, CI RESTITUISCE L'ID VOLUTO

LINEA DI COMANDO

PHP

JAVA

FACCIAMOLO DA LINEA DI COMANDO

1 Connettiamoci al database mysql su cui vogliamo operare

```
mysql -uuser -ppassword database
```

2 Dal prompt dei comandi di MySQL digitiamo la query che ci interessa

```
mysql> select distinct LAST_INSERT_ID() from istat;
```

FACCIAMOLO DA PHP

1 Il codice da inserire in file PHP è il seguente:

```
<?php
  /* Connessione e selezione del database */
  $connessione = mysql_connect("localhost",
                                   "user", "password")
      or die("Connessione non riuscita: " .
                                      mysql_error());
  mysql_select_db("ioprogrammo") or die("Selezione
                           del database non riuscita");
  /* Esecuzione di una query SQL */
  $query = "SELECT DISTINCT LAST_INSERT_ID()
                                        FROM istat;";
  $risultato = mysql_query($query) or die("Query
                         fallita: " . mysql_error() );
  $row = mysql_fetch_array($risultato);
  print $row['LAST_INSERT_ID()'];
  /* Liberazione delle risorse del risultato */
  mysql_free_result($risultato);

  /* Chiusura della connessione */
  mysql_close($connessione);
?>
```

Sostituendo ovviamente i valori di user, password, database e tabella con quelli adatti ai vostri scopi.

OTTIMIZZARE PHP

PHP mette a disposizione una funzione molto più comoda per ottenere lo stesso dato. Si tratta di *mysql_insert_id()*, che può essere banalmente usata dopo avere instaurato la connessione, con la seguente sintassi:

```
print mysql_insert_id();
```

COSA VUOL DIRE LA CLAUSOLA DISTINCT?

Utilizzando questa clausola si eliminano i valori doppi dal risultato di una query

FACCIAMOLO IN JAVA

1 Assicuriamoci di avere installato il provider per Java, lo trovate allegato al cd di ioProgrammo, oppure al sito: *http://www.mysql.com*

2 Il codice da utilizzare è il seguente:

```
package ioprogrammo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import com.mysql.jdbc.ResultSet;
public class MysqlTest
{

  /**
  * @param args
  */
  public static void main(String[] args)
  {
    // TODO Auto-generated method stub
    try
    {
      Statement stmt;
      ResultSet rs;
      Class.forName("com.mysql.jdbc.Driver")
                                    .newInstance();
      String url = "jdbc:mysql://127.0.0.1
                                    /ioprogrammo";
      Connection con = DriverManager
           .getConnection(url,"jaco", "ff19uk34");
      stmt = con.createStatement(
             ResultSet.TYPE_SCROLL_INSENSITIVE,
                   ResultSet.CONCUR_READ_ONLY);
      rs=(ResultSet)stmt.executeQuery("SELECT
        DISTINCT LAST_INSERT_ID() FROM istat;");
      rs.first();
      System.out.println(rs.getInt(
                            "LAST_INSERT_ID()"));
    }
    catch (Exception ex)
    {

      System.out.println("SQLException:
                          " + ex.getMessage());
      System.out.println("SQLState: " +
                                ex.getCause()) ;
      //System.out.println("VendorError: " +
                           ex.getErrorCode());
    }
  }
}
```

COME OTTENERE LA DIMENSIONE OCCUPATA DA UN DB MYSQL

DALLE INTERFACCE GRAFICHE È MOLTO SEMPLICE CONOSCERE LO SPAZIO OCCUPATO DALLE TABELLE E DAGLI INDICI DI UN DB, MA COME FARE DA CODICE?

Tramite l'utilizzo di un'istruzione, è possibile stabilire tali grandezze anche programmaticamente utilizzando il comando SHOW TABLE STATUS [FROM db_name] si ottiene una serie di record i quali, per ogni tabella presente nel db, forniscono tra l'altro anche le informazioni sulla grandezza dei dati e degli indici, rispettivamente nei campi Data_length e Index_length. Sommando questi valori si ottiene la dimensione del DataBase

PHP

C#

VISUAL BASIC.NET

JAVA

FACCIAMOLO DA PHP

1 Il codice è abbastanza semplice, ovviamente facciamo riferimento a un database "ioprogrammo" già creato:

```
<?php
  /* Connessione e selezione del database */
  $connessione = mysql_connect("localhost",
                                    "root", "password")
      or die("Connessione non riuscita: " .
                                         mysql_error());
  mysql_select_db("ioprogrammo") or
          die("Selezione del database non riuscita");

  /* Esecuzione di una query SQL */
  $query = "show table status";

  $risultato = mysql_query($query) or die("Query
                              fallita: " . mysql_error() );

  echo "<table>\n";

  while ($linea = mysql_fetch_array($risultato,
                                   MYSQL_ASSOC)) {
      echo "\t<tr>\n";
      echo "Name: $linea[Name]";
      echo " Data Length: $linea[Data_length]".
            "-".round(($linea["Data_length"]/1024)
                                     /1024, 5)."Mb";
      echo " Index Length: $linea[Index_length]".
            "-".round(($linea["Data_length"]/1024)
                                /1024, 5)."Mb<br>";
      echo "\t</tr>\n";
      $size = $size + $linea["Data_length"]
                           +$linea["Index_length"];
  }

  print "</table>\n";

  echo "Totale=".round(($size/1024)/1024,2);

  /* Liberazione delle risorse del risultato */
  mysql_free_result($risultato);

  /* Chiusura della connessione */
  mysql_close($connessione);
?>
```

COME FUNZIONA

Viene stabilita una connessione al server e poi selezionato il database "ioprogrammo". Viene eseguita una query "Show table status" attraverso "mysql_query". Il resultset ottenuto viene scansionato tramite un ciclo di while. Nel ciclo in questione di volta in volta si aggiorna una variabile size con i valori contenuti in data_length ed index_length. Contemporaneamente vengono stampati a schermo i valori singoli divisi per tabella.
La divisione per 1024 e poi ancora per 1024 serve a passare da Byte a MB.

FACCIAMOLO CON C#

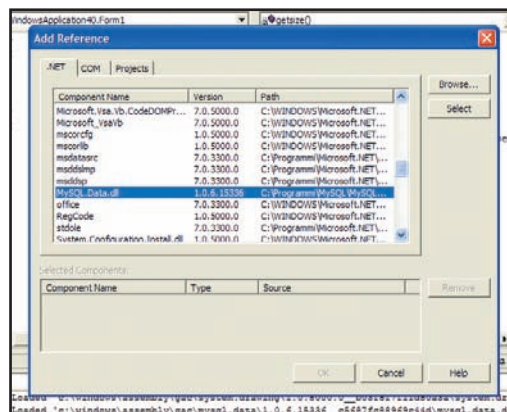
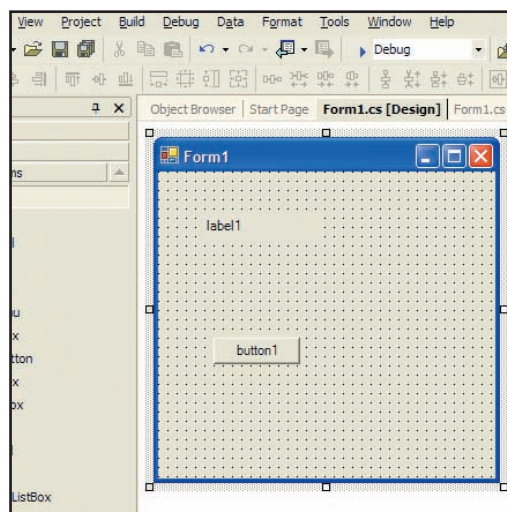
Prima di tutto assicuriamoci di avere installato il provider .NET per MySql. Lo trovate nel cdrom allegato alla rivista

1 Cliccate con il tasto destro del mouse nel solution explorer alla voce "References" e nel menu a tendina che segue su *"Add Reference"*.

I NAMESPACE DA IMPORTARE IN C#

È necessario aggiungere all'inizio del file di gestione del codice la riga

```
using MySql.Data
.MySqlClient;
```

2 Nella finestra che segue individuate la voce MySQLData.Dll e ciccate due volte su di essa, infine su “Ok”

3 Nella form aggiungiamo un pulsante e una label, trascinandoli dalla toolbox

I NAMESPACE DA IMPORTARE IN VB .NET

All'inizio del codice è necessario aggiungere:

```
Imports MySql.Data
       .MySqlClient
```

4 Clicchiamo due volte sul bottone per fare genere il template del codice che gestirà l'evento “OnClick”. Il codice da inserire è il seguente:

```
private void button1_Click(object sender,
                              System.EventArgs e)
{
   label1.Text=getsize().ToString();
}
```

5 Scriviamo il metodo getsize, aggiungendolo poco prima del metodo che gestisce l'evento OnClick. Il codice da inserire sarà il seguente:

```
long getsize(){
   string ConnString="Database=mailservice;Data
                     Source=localhost;User Id=root;
                                Password=atp0h21";
   MySqlConnection Connessione = new
                        MySqlConnection(ConnString);
   Connessione.Open();
   MySqlCommand Cmd =
                   Connessione.CreateCommand();
   Cmd.Connection = Connessione;
   Cmd.CommandText = "  SHOW TABLE STATUS";
   MySqlDataReader Reader= Cmd.ExecuteReader();
   long size=0;
   while (Reader.Read())
   {
      size+=long.Parse(Reader["Data_length"
                                 ].ToString());
      size+=long.Parse(Reader["Index_length"
                                 ].ToString());
   }
   Connessione.Close();
   return size;
}
```

FACCIAMOLO CON VISUAL BASIC.NET

I passi dall'uno al tre rimangono identici

1 Clicchiamo due volte sul bottone e aggiungiamo il codice di gestione dell'e vento OnClick, che diventa:

```
Private Sub Button1_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
                            Handles Button1.Click
   Label1.Text = getsize().ToString
End Sub
```

2 Esplicitiamo il metodo getisize() che in Visual Basic.NET diventa

```
Private Function getsize() As Long
   Dim ConnString As String = "Database=
        mailservice;Data Source=localhost;User
                    Id=root;Password=atp0h21"
   Dim Connessione As MySqlConnection = New
                     MySqlConnection(ConnString)
   Connessione.Open()
   Dim Cmd As MySqlCommand =
                  Connessione.CreateCommand()
   Cmd.Connection = Connessione
   Cmd.CommandText = "  SHOW TABLE STATUS"
   Dim Reader As MySqlDataReader =
                          Cmd.ExecuteReader()
   Dim size As Long = 0
   Do While Reader.Read
      size = size + Long.Parse(Reader(
                  "Data_length").ToString())
      size = size + Long.Parse(Reader(
                 "Index_length").ToString())
   Loop
   Connessione.Close()
```

```
        Return size

    End Function
```

FACCIAMOLO DA JAVA

1 Assicuratevi di avere installato il connector per Java. Lo trovate nel cd allegato ad ioProgrammo, oppure all'indirizzo: *http://dev.mysql.com /downloads/connector/j/3.1.html*. Inserite il jar mysql-connector-java-3.1.11-bin.jar in una directory contenuta nel classpath. Il codice da inserire è il seguente

```
package ioprogrammo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import com.mysql.jdbc.ResultSet;
public class MysqlTest
{
  /**
  * @param args
  */
  public static void main(String[] args)
  {
    // TODO Auto-generated method stub
    try
    {
      Statement stmt;
      ResultSet rs;
      Class.forName("com.mysql.jdbc.Driver"
                      ).newInstance();
      String url = "jdbc:mysql:
              //127.0.0.1/ioprogrammo";
      Connection con = DriverManager
          .getConnection(url,"root", "password");
      stmt = con.createStatement(
          ResultSet.TYPE_SCROLL_INSENSITIVE,
              ResultSet.CONCUR_READ_ONLY);
      rs=(ResultSet)stmt.executeQuery(
                  "SHOW TABLE STATUS");
      double totale = 0;
      while(rs.next()){
        totale += rs.getInt("Data_length"
            )+rs.getInt("Index_length"); }
      System.out.println((totale/1024)/1024);
    }
    catch (Exception ex) {
      System.out.println("SQLException: " +
                      ex.getMessage());
      System.out.println("SQLState: " +
                      ex.getCause()) ;
      //System.out.println("VendorError: " +
                      ex.getErrorCode());
    } }
}
```

COME FUNZIONA

Con l'istruzione Class.forName("com.mysql.jdbc .Driver").newInstance(); il driver per mysql viene registrato all'interno del programma, rendendone possibile l'utilizzo. L'oggetto con di classe connection viene instanziato e il suo costruttore inizializzato grazie al metodo getConnection. Questo metodo genera un'eccezione se la connessione non va a buon fine. Per eseguire la query si usa un oggetto statement. Il risultato della query viene salvato in un oggetto di tipo resultset che può facilmente essere scandito con un loop.

NOTA

IN AMBIENTE LINUX

Se state provando in locale è importante che nel file /etc/hosts l'ordine degli host sia:

```
127.0.0.1 localhost
      localhost.localdomain
```

altrimenti potreste avere problemi nella gestione dei permessi

COSA È LA QUERY CACHE IN MYSQL

Per migliorare le prestazioni delle query in mysql, è possibile utilizzare una cache integrata. Tale cache è un meccanismo che mantiene in memoria i risultati di una query, permettendo così di velocizzare le susseguenti query simili. Lo stato della funzionalità di query cache può assumere tre valori: OFF =0, ON=1, ON DEMAND=2. Per scegliere tale valore si deve agire sul valore di configurazione di mysql.ini:

```
query-cache-type = [0, 1 o 2]
```

volendo si può anche scegliere la grandezza della cache impostando i valore query-cache-size = XM con X il numero di megabytes desiderato.
Se il valore di query-cache-type è 0, non si avrà la cache.
Se è 1, la cache sarà attivata per ogni query. Occorre ricordare che le query vengono immagazzinate in maniera case sensitive e che per avere l'incremento delle prestazioni desiderato il resultset dovrà essere più piccolo della memoria cache impostata.
Si ricordi che le query di INSERT, UPDATE, DELETE, TRUNCATE, ALTER TABLE, DROP TABLE, DROP DATABASE rimuovono le query dalla cache.
Tale risultato si può ottenere anche col comando RESET QUERY CACHE.
Se il valore di query-cache-type è 2 si ha l'inserimento della query in cache solamente quando richiesto (ON DEMAND), tramite l'istruzione:

```
SELECT SQL_CACHE  * FROM MiaTabella.
```

Per vedere lo stato della cache, si posso utilizzare le istruzioni:

```
SHOW VARIABLES LIKE '%query_cache%'
```

e

```
SHOW status LIKE '%cache%'
```

COME CREARE UNA TABELLA CON MYSQL

CI SONO VARI METODI. A LINEA DI COMANDO, PER MEZZO DI INTERFACCE GRAFICHE, IN MODO PROGRAMMATICO DA PHP O DA ALTRI LINGUAGGI. VEDIAMO UN ESEMPIO IN VARI CASI

LINEA DI COMANDO

PHP

MYSQL

FACCIAMOLO DA LINEA DI COMANDO

1 Presupposto che mysql.exe sia incluso nel path di sistema, creiamo un file di testo composto come segue:

```
CREATE TABLE MiaTabella (
IDImpiegato INTEGER UNSIGNED NOT NULL
AUTO_INCREMENT PRIMARY KEY,
Nome VARCHAR(45) NOT NULL,
AnnoNascita INTEGER UNSIGNED NOT NULL,
Note TEXT
);
```

chiamiamo questo file ad esempio *tabella.sql*

2 Apriamo il prompt dei comandi se siamo in ambiente Windows oppure la shell se siamo in ambiente linux e digitiamo:

```
mysql -uuser -ppassword database < tabella.sql
```

dove user è un utente abilitato ad accedere a mysql, password è la password per l'utente in questione, database è il database dentro cui volete creare la tabella.

CREARE UN DATABASE

Per creare un database in MySQL, da linea di comando potete usare la seguente sintassi

```
mysqladmin -uroot
-ppassword create
database
```

dove password è la password dell'utente root

COME FUNZIONA

Il meccanismo sfrutta il simbolo di redirezione "<" che preleva il contenuto del file tabella.sql e lo esegue come istruzione all'interno di mysql. poiché nel file in questione abbiamo messo proprio i comandi per creare una tabella, Ecco che otteniamo il risultato voluto.

FACCIAMOLO DA PHP

1 Il codice da utilizzare è il seguente:

```
<?php
/* Connessione e selezione del database */
$connessione = mysql_connect("localhost",
"user", "password")
or die("Connessione non riuscita: " .
mysql_error());
print "Connesso con successo";
mysql_select_db("ioprogrammo") or
die("Selezione del database non riuscita");

/* Esecuzione di una query SQL */
$query = "
CREATE TABLE MiaTabella (
IDImpiegato INTEGER UNSIGNED NOT NULL
AUTO_INCREMENT PRIMARY KEY,
Nome VARCHAR(45) NOT NULL,
AnnoNascita INTEGER UNSIGNED NOT NULL,
Note TEXT
);
";
$risultato = mysql_query($query) or die(
"Query fallita: " . mysql_error() );

/* Liberazione delle risorse del risultato */
mysql_free_result($risultato);

/* Chiusura della connessione */
mysql_close($connessione);
?>
```

COME FUNZIONA

La funzione mysql_connect fornisce le credenziali d'accesso e si connette al database. Se la connessione non riesce il programma viene terminato con una funzione "die" e restituisce un apposito messaggio d'errore, viceversa si passa alla funzione mysq_connect che seleziona fra i database di mysql quello desiderato.
La query di creazione della tabella viene inglobata in una stringa che a sua volta viene passata come parametro a mysql_query che appunto l'esegue. Infine vengono liberate le risorse per la connessione.

CREARE UN UTENTE CON PASSWORD

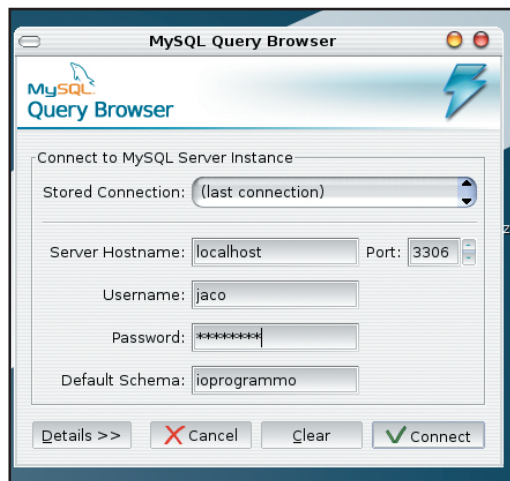
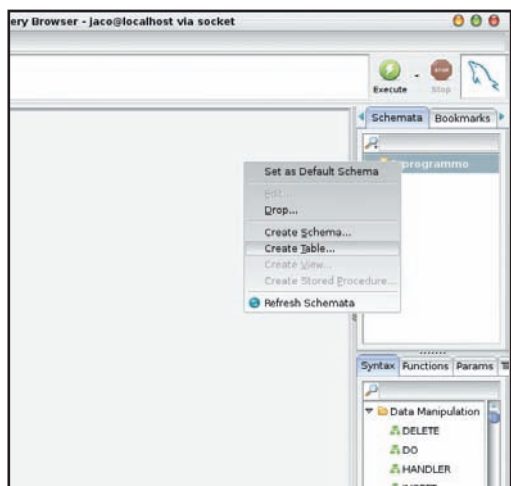
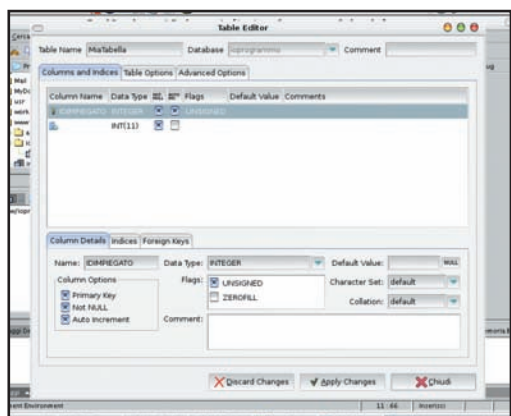
Per creare un utente abilitato ad accedere a un database, potete usare la seguente sintassi

```
mysql -uroot -ppassword
mysql> grant all privileges on
database.* to 'user'@'localhost' identified by 'password';
```

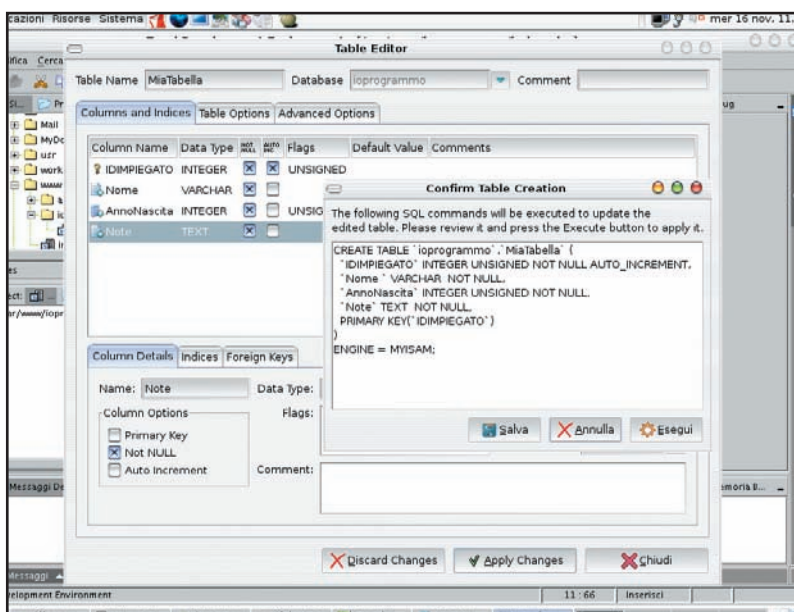
dove user è il nome dell'utente che volete creare, database è il nome del database a cui l'utente deve avere un accesso completo, password è la password che volete assegnargli.

FACCIAMOLO DA MYSQL QUERY BROWSER

1 Inseriamo i parametri di connessione necessari ad utilizzare il database. In particolare, il nome

del server a cui connetterci: localhost se si tratta di un server installato sulla macchina dove stiamo eseguendo il test. Il nome utente e la password. Nel campo schema inseriamo il nome del database a cui connettersi

2 Clicchiamo com il tasto destro del mouse sul nome indicante il database e di seguito nel menu a tendina su "create table".

3 Inseriamo i vari campi che compongono la nostra tabella, utilizzando le facilities messe a disposizione dall'interfaccia.

COME POSSO MIGRARE A MYSQL DA ACCESS?

Sul sito di MySQL all'indirizzo http://www.mysql.com/products/tools/migration-toolkit/ oppure nel cdrom di ioProgrammo trovate il MySQL Migration Toolkit. Si tratta di un tool che fornisce un wizard per la migrazione rapida da altri database a MySQL. I database supportati sono davvero tanti e comprendono Access, MS Sql Server ed Oracle. Il Wizar è abbastanza semplice ed intuitivo e la migrazione nei casi che abbiamo sperimentato non è stata particolarmente difficoltosa.

4 Quando siamo pronti clicchiamo su "apply changes" e di seguito su "esegui".

I VARI TIPI DI TABELLA

Durante la creazione della tabella, è possibile specificare come direttiva finale un engine.

Ad esempio:

```
Create table 'ioprogrammo.test' (
...
)
engine = MYISAM
```

L'engine specifica un tipo di tabella. In particolare i due engine più utilizzati sono MYISAM e INNODB, ognuno dei due engine offre a MYSQL particolari vantaggi. Ad esempio con MYISAM è possibile utilizzare la ricerca full text oppure le stored procedure del nuovo MySQL5.

Con InnoDB è possibile sfruttare le transazioni, che risultano indispensabili in software che necessitano un accesso concorrente ai dati.

Un elenco dei vari engine supportati da MySQL è disponibile alla URL http://dev.mysql.com/doc/refman/5.0/en/storage-engines.html

OTTENERE LA DIMENSIONE DI UN'IMMAGINE

SUPPONETE DI VOLER REALIZZARE UNA GALLERY DI FOTOGRAFIE, E PER CIASCUNA IMMAGINE STAMPARE SOTTO DI ESSA LA SUA DIMENSIONE IN PIXEL E LA SUA DIMENSIONE IN BYTE, ECCO COME FARE

JAVA SCRIPT

PHP

ASP.NET

DIFFERENZE TRA JAVASCRIPT E PHP

Mentre nel caso di JavaScript vengono restituite le dimensioni in larghezza e altezza definite dall'utente all'interno dei tag width e height, nel caso di PHP vengono invece restituite le dimensioni reali dell'immagine.

I NAMESPACE DA IMPORTARE

È necessario usare il namespace System.IO inserendo una riga

Using System.IO

all'inizio del codice

FACCIAMOLO CON JAVASCRIPT

1 La pagina HTML base assomiglierà a qualcosa del genere:

```
<body>
<p><img id=ninfea src="../Ninfee.jpg" alt="s" width=
                    "286" height="234" /> +</p>
<p> </p>
```

2 Il codice javascript che va inserito immediatamente sotto l'immagine è il seguente:

```
<script language="javascript">

   document.write("Larghezza immagine (attributo
          width) = " + document.images[0].width) ;
   document.write("<br/>");
   document.write("Altezza immagine (attributo
          height) = " + document.images[0].height) ;
   document.write("<br/>");
   document.write("Dimensione in byte immagine
          (fileSize) = " + document.images[0].fileSize) ;

</script>

</body>
```

COME FUNZIONA

È abbastanza semplice. Il codice JavaScript accede alle proprietà dell'immagine indicizzata con il numero 0, e stampa a video con l'istruzione document.write le informazioni dovute. Ovviamente le immagini successive avranno un indice incrementale.

FACCIAMOLO IN PHP

1 Prima di tutto assicuriamoci che l'estensione GD sia abilitata nel php.ini. Tipicamente il php.ini è collocato in c:\windows. Editatelo e controllate che la seguente riga:

```
extension=php_gd2.dll
```

Non sia preceduta da un punto e virgola.

2 Il codice da inserire per fare apparire l'immagine è identico a quello del passo 1 di Java Script.

Per ottenere le dimensioni è invece opportuno inserire il seguente codice:

```
<?php
list($width, $height, $type, $attr) =
                         getimagesize("../ninfee.jpg");
echo "Larghezza: ".$width."<br>";
echo "Altezza: ".$height."<br>";
echo "Dimensioni: ".filesize("../ninfee.jpg");
?>
```

COME FUNZIONA

Prima di tutto viene sfruttata la funzione getImageSize delle GD che restituisce appunto le dimensioni in larghezza e altezza dell'immagine. Per le dimensioni del file si fa invece affidamento alla funzione filesize che non appartiene alle GD ed è inclusa invece nel linguaggio .PHP

FACCIAMOLO IN ASP.NET CON C#

1 Nella webform inseriamo tre label e l'immagine di cui vogliamo ottenere le informazioni. Inseriremo il codice per recuperare altezza, larghezza e dimensione nell'evento onload della pagina.
Come segue:

```
private void Page_Load(object sender,
                              System.EventArgs e)
  {
     Bitmap img = new Bitmap(Server.MapPath(
            "/WebApplication2")+@"\..\ninfee.jpg");
     FileInfo f = new FileInfo(Server.MapPath(
            "/WebApplication2")+@"\..\ninfee.jpg");
     Label1.Text=img.Width.ToString();
     Label2.Text=img.Height.ToString();
     Label3.Text=f.Length.ToString();
  }
```

FACCIAMOLO IN ASP.NET CON VISUAL BASIC

1 Anche in questo caso avremo bisogno di tre label sulla webform, opzionalmente dell'immagine di cui vogliamo recuperare le informazioni.
Anche in questo caso inseriremo il codice di ge-

stione nell'evento load della pagina, come segue:

```
Private Sub Page_Load(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                                    Handles MyBase.Load
        Dim img As Bitmap = New
                              Bitmap(Server.MapPath(
                   "/WebApplication2") + "\..\ninfee.jpg")
        Dim f As FileInfo = New FileInfo(
                    Server.MapPath("/WebApplication2")
                                     + "\..\ninfee.jpg")
        Label1.Text = img.Width.T
oString()
        Label2.Text = img.Height.ToString()
        Label3.Text = f.Length.ToString()
    End Sub
```

COME POSSO COPIARE IL CONTENUTO DI UNA TEXTBOX NELLA CLIPBOARD?

Il codice che segue può essere utilizzato per effettuare i classici Copia e Incolla dalla memoria utilizzando javascript. La cosa può essere utile quando si voglia, ad esempio, dare la possibilità di copiare spezzoni di codice abbastanza lunghi da aree di testo su una pagina web, senza dover selezionare tutto il contenuto dell'area, con il rischio di dimenticarne qualche pezzo "nella coda del mouse". Attenzione che il codice funziona solo e soltanto su Internet Explorer, in quanto la proprietà utilizzata "clipboardData" è specifica del browser Microsoft.

```
<html>
  <head>
    <title>Copia e incolla da clipboard
        SOLO PER INTERNER EXPLORER</title>
  </head>
  <body>
  <form name="frmCopiaIncolla"
                    id="frmCopiaIncolla">
   Copia:
     <textarea cols="40" rows="10" name=
                  "txtCopia" id="txtCopia">
      Questo testo verra' copiato nella
                               clipboard.
      Premi il tasto copia a fianco.
     </textarea>
        <input type="button" value=
                        "Copia" onClick=
                   "CopyToClipboard()">
     <br>
     Incolla:<textarea cols="40" rows="10"
          name="txtIncolla" id="txtIncolla">
</textarea>
     <input type="button" value="Incolla"
         onClick="PasteFromClipboard()">
</form>
  </body>
<script language="javascript">
 function CopyToClipboard()
   {
     window.clipboardData.setData("Text",
       document.getElementById('txtCopia'
                              ).value);
   }
          function PasteFromClipboard()
   {
   document.getElementById('txtIncolla')
        .value=window.clipboardData.getData
                              ( "Text");
   }
   </script>
</html>
```

È chiaro che una volta che avete copiato il testo in memoria, lo potete incollare dove volete, ad esempio sul vostro editor di testo. Il tasto "Copia" effettua la copia del contenuto della textarea txtCopia in memoria, mentre il tasto "Incolla" trasferisce il contenuto della memoria (qualunque esso sia) nella casella di testo txtIncolla.

COME CREARE UNA FORM CON DUE SUBMIT

SUPPONIAMO DI AVERE UNA PAGINA WEB CON UN'UNICA CASELLA DI TESTO ALL'INTERNO DELLA QUALE SCRIVERE UNA FRASE DA RICERCARE. NELLA STESSA PAGINA ABBIAMO DUE BOTTONI. VOGLIAMO CHE CLICCANDO SUL PRIMO SI ESEGUA UNA RICERCA SU GOOGLE, CLICCANDO SUL SECONDO UNA RICERCA SU ALTAVISTA. COME FARE?

FACCIAMOLO CON JAVASCRIPT

1 Il codice che gestisce la form è il seguente:

```
<form name="frmMultiSubmit" id=
                "frmMultiSubmit" target="_blank">
   Ricerca:<input type="text" id="q" name="q">
      <input type="button" value="Cerca con
        Altavista" onClick="Sottometti('Altavista')">
      <input type="button" value="Cerca con Google"
                  onClick="Sottometti('Google')">
     </form>
```

2 Dobbiamo implementare la function "sottometti" in javascript.

JAVA SCRIPT

C#

ASP.NET

QUESTIONE DI TARGET

Si potrebbe inserire un terzo bottone premendo il quale si potrebbero aprire due finestre che contemporaneamente fanno la ricerca sui due motori. Il bottone in questione potrebbe richiamare una funzione javascript "tutti" che contiene il seguente codice:

```
function Tutti()
{   document.frmMultiSubmit.action =
        "http://it.altavista.com/web/results";
    document.frmMultiSubmit.submit();
    document.frmMultiSubmit.action =
            "http://www.google.it/search";
    document.frmMultiSubmit .submit();
}
```

Si noti che il target della form è "target="_blank" e questo consente di aprire due finestre separate.

Una possibile implementazione è la seguente:

```
<script language="javascript">
   function Sottometti(dove)
  {
     switch(dove)
     {
        case 'Altavista':
        var strURL = "http://it.altavista.com/web
                                        /results";
        break
        case 'Google':
        var strURL = "http://www.google.it/search";
        break
        default:
        alert("Caso non previsto !");
        exit;
     }
     document.frmMultiSubmit.action = strURL;
     document.frmMultiSubmit.submit();
}

<script language="javascript">
   document.write("Larghezza immagine (attributo
          width) = " + document.images[0].width) ;
   document.write("<br/>");
   document.write("Altezza immagine (attributo
          height) = " + document.images[0].height) ;
   document.write("<br/>");
   document.write("Dimensione in byte immagine
        (fileSize) = " + document.images[0].fileSize) ;

</script>

</body>
```

IN AMBIENTE LINUX

Se state provando in locale è importante che nel file /etc/hosts l'ordine degli host sia:

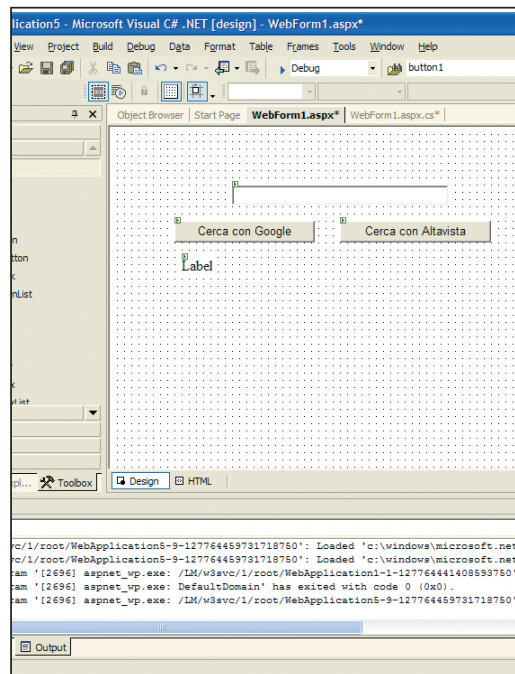
127.0.0.1 localhost
localhost.localdomain

altrimenti potreste avere problemi nella gestione dei permessi

COME FUNZIONA

Quando l'utente clicca su uno dei due bottoni viene invocata la funzione "sottometti" che prende in input una stringa. A seconda del bottone premuto viene passata come parametro la stringa "google" oppure la stringa "altavista". Con uno switch si agisce sulla property action della form e si invoca subito dopo il submit.

FACCIAMOLO IN ASP.NET CON C#

1 Il nostro progetto avrà bisogno di tre bottoni, una label e una textbox. Trasciniamoli dalla toolbox sulla web form

2 Cliccando due volte sul primo bottone facciamo generare il template di gestione dell'evento "OnClick". Il codice da inserire è il seguente:

```
private void Button1_Click(object sender,
                           System.EventArgs e)
{
   if (TextBox1.Text != "")
   {
      Response.Redirect("http://it.altavista.com
          /web/results?q="+System.Web.HttpUtility
                     .UrlEncode(TextBox1.Text));
   }
   else
   {
      Label1.Text  = "Devi inserire una parola";
   }
}
```

3 Allo stesso modo cliccando sul secondo bottone generiamo il codice di gestione dell'evento OnClick. Il codice da inserire è il seguente:

```
private void Button2_Click(object sender,
                           System.EventArgs e)
{
   if (TextBox1.Text != "")
   {
```

```
    Response.Redirect("http://www.google.it
        /search?q="+System.Web.HttpUtility.UrlEncode(
                                        TextBox1.Text));
  }
  else
  {
     Label1.Text  = "Devi inserire una parola";
  }
}
```

FACCIAMOLO IN ASP.NET CON VISUAL BASIC

1 I passi fino al secondo punto sono identici, eccetto per il codice da inserire per la gestione dell'evento onclick, che diventa:

```
Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                                Handles Button1.Click
        If Not (TextBox1.Text = "") Then

            Response.Redirect("http://it.altavista.com
            /web/results?q=" + System.Web.HttpUtility
                            .UrlEncode(TextBox1.Text))
        Else
```

VALIDAZIONE DELLA FORM

Questa volta abbiamo inserito un codice di controllo, per gestire la validazione dell'input. Se l'utente non inserisce testo all'interno della casella, viene generato un apposito messaggio d'errore. Per contro non è più possibile gestire il target. Responce.redirect infatti non prevede alcun parametro per la gestione del target

```
            Label1.Text = "Devi inserire una parola"
        End If
    End Sub

    Private Sub Button2_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                                Handles Button2.Click
        If Not (TextBox1.Text = "") Then
    Response.Redirect("http://www.google.it
    /search?q=" + System.Web.HttpUtility.UrlEncode(
                                        TextBox1.Text))

        Else
            Label1.Text = "Devi inserire una parola"
        End If

    End Sub
```

COME POSSO CONTROLLARE CHE I CAMPI DI UNA FORM SIANO RIEMPITI?

Nel caso in cui vogliate verificare lato client i dati inseriti in un form, prima di sottometterli potete usare un procedimento come quello che segue. In questo caso, si verifica di aver inserito un valore non nullo all'interno della stringa prima di effettuare la sottomissione al motore di ricerca.
È chiaro che i controlli formali si possono "complicare" a piacere.

```
<html>
  <head>
    <title>Form con controlli formali prima
                    della sottomissione</title>
  </head>
  <body>
    <form name="frmVerifica" id=
                "frmVerifica" target="_blank">
        Ricerca:<input type="text" id="q"
                                name="q">
        <input type="button" value="Cerca
            con Google" onClick="Verifica()">
    </form>
  </body>
  <script language="javascript">
    function Verifica()
    {
      if(document.getElementById('q')
                                .value=="")
          alert("Valorizzare il campo di
                                Ricerca !");
      else
      {
          document.frmVerifica.action =
              "http://www.google.it/search";
          document.frmVerifica.submit();
      }
    }

  </script>
</html>
```

PERCHÉ USARE URLENCODE?

Alcuni caratteri presenti nella textbox potrebbero non essere accettati dal motore che riceve la richiesta. Ad esempio uno spazio bianco o un punto potrebbe causare malfunzionamenti.
Utilizzando la funzione urlencode ci assicuriamo ad esempio che gli spazi bianchi siano convertiti in "%20" universalmente riconosciuto dal protocollo http per quanto riguarda la gestione delle query string.

L'XML è servito con java e StaX

Una panoramica delle tecnologie XML per Java. Introdurremo le API StAX, capiremo come estrarre da un documento XML elementi ed attributi e realizzeremo nuovi documenti XML

J2SE

REQUISITI

Conoscenze richieste
Linguaggio Java

Software
Java2 SDK 1.4, Eclipse 3.1

Impegno

Tempo di realizzazione

La possibilità di accedere a dati in formato XML è una caratteristica indispensabile per una piattaforma moderna. Java dispone di diverse tecnologie per accedere a questo tipo di dati, a cui si è aggiunto ultimamente il parsing di tipo "*pull*". Lo scopo di questa nuova tecnologia è quello di semplificare le applicazioni e nello stesso tempo migliorare le prestazioni.
Per mettere nella giusta prospettiva questa tecnologia è però necessario riassumere le altre API presenti nella piattaforma.

SUPPORTO XML

La piattaforma Java dispone di molte possibilità per trattare con dati XML, sia semplici, come normali documenti che devono essere letti o scritti, oppure complessi, come ad esempio le tecnologie legate ai Web Service e SOAP. Le tecnologie principali per il parsing e produzione di documenti XML sono:

- **DOM** *(Document Object Model)*. Con queste API, molto diffuse e disponibili anche in altri linguaggi di programmazione, è possibile navigare la struttura gerarchica di un documento XML in modo facile. Uno svantaggio è la lentezza, in quando il documento deve essere letto integralmente in fase di parsing prima di poter essere usato. Un altro svantaggio è la grande occupazione di memoria, in quanto il parser crea un modello in memoria di tutto il documento, se questo è molto grande è necessario molto spazio.
- **SAX** *(Simple API for XML processing)*. I vantaggi di questa tecnologia sono la velocità e la bassa occupazione di memoria. Il parser si occupa di decodificare il documento XML e per ogni elemento riscontrato genera un evento che l'applicazione deve intercettare e gestire. Lo svantaggio è che richiede molto codice di supporto e che questo non è molto orientato agli oggetti. Inoltre, non è possibile modificare il contenuto del file, a meno di riscriverlo da zero.
- **DOM4J** *(http://dom4j.org/)*. Questa API è molto performante e ben integrata con il linguaggio Java. Offre interfacce DOM e SAX ma ha lo svantaggio di non essere una API "ufficiale". È sostanzialmente un componente open source con una interfaccia diversa;
- **JAXB** *(Java Architecture for XML Binding)*. Questa tecnologia va ben oltre la semplice lettura/scrittura di documenti XML. Fornisce infatti una mappatura completa tra schemi di documenti XML ed il linguaggio Java. Un apposito strumento si occupa di generare il codice necessario ad eseguire il parsing e la produzione dello schema XML di riferimento. Il programmatore avrà dunque solo a che fare con normali classi Java, senza dover entrare nel dettaglio della tecnologia XML.

La tecnologia *StAX (Streaming API for XML)* si pone tra DOM e SAX, cercando di prendere il meglio da entrambi i mondi.

COME INIZIARE

Per eseguire gli esempi è necessario scaricare una implementazione di StAX, ad esempio quella presente all'interno del JWSDP, ottenibile all'indirizzo http://java.sun.com/webservices/downloads/webservicespack.html

INTRODUZIONE A STAX

Con i parser *StAX*, il controllo delle operazioni rimane in mano all'applicazione, come nel caso

di DOM, ma senza gli svantaggi di quest'ultima tecnologia. In *StAX* la lettura ed interpretazione del file XML è infatti eseguita serialmente, in modo similare a SAX. La mancanza di un modello dei dati in memoria riduce la quantità di memoria richiesta per la decodifica del documento XML. Lo svantaggio principale del modello StAX è la minor disponibilità di funzionalità rispetto a DOM o SAX. Tra i vantaggi si trovano:

- **semplicità.** StAX è ancora più semplice di SAX e DOM. Implementa un parser di tipo *"pull"*, e fornisce l'accesso al documento XML utilizzando API basate su iteratori;
- **familiare.** L'utilizzo degli iteratori è un approccio frequentemente utilizzato nelle librerie di base di Java ed in quelle aggiuntive. Il loro uso dovrebbe essere quindi familiare allo sviluppatore;
- **lettura/scrittura.** Con StAX è possibile leggere e scrivere documenti in modo più efficiente rispetto ad altre API come DOM;
- **prestazioni.** In modo similare a SAX, StAX ha ottime prestazioni e richiede poca memoria, in quanto non costruisce un modello dei dati XML in memoria.

Le API *StAX* (strette cugine di SAX, con cui non bisogna fare confusione), sono indicate per documenti XML di grandi dimensioni, dove le prestazioni e la memoria possono diventare un problema. Sono altresì indicate per programmi semplici, dove il ricorso ad API più complesse può sembrare eccessivo. Per lavorare con StAX è necessario installare nel sistema una sua implementazione.
Ad oggi la piattaforma Java non include StAX nelle API *JAXP* (*Java API for XML Processing*), che offrono solo DOM e SAX. Quindi l'installazione di base dell'ambiente di sviluppo di Java non è sufficiente per utilizzare questa tecnologia. È disponibile però all'interno del *Java Web Services Developer Pack*, un pacchetto di tecnologie per l'utilizzo dei servizi Web in Java che include anche API collegate.

LEGGERE UN FILE XML

Per cominciare a capire il funzionamento di StAX, vediamo come leggere un semplice documento XML che contiene i dati di un ordine inviato attraverso Internet. Nella sezione destinatario sono presenti i dati relativi a chi ha ordinato la merce, mentre ciascun elemento prodotto contiene i singoli articoli acquistati. Il documento ha la struttura seguente:

```
<?xml version="1.0"?>
<ordine id="9938277378" data="20051201"
                                  utente="AB01392">
  <destinatario>
    <nome>Mauro</nome>
    <cognome>Gallipari</cognome>
    <via>Via Lombardia</via>
    <numero>1/2</numero>
    <cap>20100</cap>
    <localita>Milano</localita>
    <provincia>MI</provincia>
  </destinatario>
  <prodotto id="772666253">
    <descrizione>Disco orario solare</descrizione>
    <quantita>3</quantita>
    <prezzo>5,99</prezzo>
  </prodotto>
  <prodotto id="776747374">
    <descrizione>Pupazzo scimmietta</descrizione>
    <quantita>1</quantita>
    <prezzo>13,99</prezzo>
  </prodotto>
  <prodotto id="771826531">
    <descrizione>Maglietta "mi sento fortunato"
                                    </descrizione>
    <quantita>2</quantita>
    <prezzo>15,99</prezzo>
    <dimensione>M</dimensione>
  </prodotto>
  <prodotto id="775425434">
    <descrizione>Lampada USB "On Air"
                                    </descrizione>
    <quantita>1</quantita>
    <prezzo>12,99</prezzo>
  </prodotto>
</ordine>
```

La classe che stampa il destinatario importa una serie di classi. Quelle relative a StAX sono presenti nel package *javax.xml.stream:*

```
package it.edmaster.ioprogrammo.stax;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLStreamException;
import javax.xml.stream.XMLStreamReader;
import javax.xml.stream.events.XMLEvent;
/**
 * StampaDestinatario
*/
public class StampaDestinatario
{
  /**
   * @param args
```

J2SE

NOTA

LE ORIGINI DI STAX

Queste API sono state studiate inizialmente da BEA Systems, un grosso produttore software per Java e si può rintracciare ancora documentazione in merito sul loro sito per sviluppatori: http://dev2dev.bea.com /xml/stax.html. È anche possibile scaricare la specifica implementazione presente nel loro application server WebLogic.

I TUOI APPUNTI

```
 * @throws FileNotFoundException
 * @throws XMLStreamException
 */
 public static void main(String[] args) throws
          FileNotFoundException, XMLStreamException
{
```

Il file che contiene l'ordine è *ordine.xml*. Il programma crea anche una serie di variabili che saranno utilizzate per ospitare le informazioni sul soggetto:

```
String filename = "ordine.xml";
String nome = "";
String cognome = "";
String via = "";
String numero = "";
String cap = "";
String localita = "";
String provincia = "";
```

Il primo passo per eseguire il parsing StAX è quello di creare una factory di oggetti *XMLStreamReader* attraverso il metodo *newInstance():*

```
XMLInputFactory factory =
                    XMLInputFactory.newInstance();
```

Il file di input viene letto attraverso una classe standard della piattaforma Java. *FileInputStream* si aspetta nel costruttore il nome del file da leggere:

```
FileInputStream fis = new FileInputStream(filename);
```

A questo punto è possibile creare il parser XML utilizzando il metodo *createXMLStreamReader()* passando il nome del file come riferimento e l'oggetto *FileInputStream* da utilizzare per la lettura:

```
XMLStreamReader reader =
   factory.createXMLStreamReader(filename, fis);
```

L'oggetto ottenuto, *reader,* può essere ora utilizzato per leggere gli elementi del file, utilizzando i metodi classici degli iteratori. Il programma infatti crea un ciclo *while* che continua fino a che il metodo *hasNext()* dell'oggetto *reader* torna *true*. Questo significa che non sono stati letti ancora tutti gli elementi del file.
Ad ogni chiamata del metodo *hasNext()* nell'oggetto reader sarà presente il nodo XML successivo:

```
while (reader.hasNext()) {
```

Una informazione importante sul nodo XML attuale è il tipo di evento, che permette di capire se si è di fronte all'inizio del documento, alla sua fine, all'inizio di un elemento e così via:

```
int type = reader.getEventType();
```

Ad esempio, nel nostro caso all'inizio del documento vengono stampate alcune informazioni di introduzione:

```
switch( type ) {
    case XMLEvent.START_DOCUMENT:
    System.out.println("Destinatario:");
    System.out.println("====");
    break;
```

Ma l'evento più importante è sicuramente l'inzio di un elemento, perché permette di estrarne il relativo valore. Il nome dell'elemento in oggetto si ottiene tramite il metodo *getLocalName()*, che ritorna valori solo se il metodo *hasName()* ritorna *true*. Questo meccanismo è necessario perché non tutti gli elementi hanno per forza un nome, si pensi ad esempio agli elementi di testo:

```
case XMLEvent.START_ELEMENT:
if (reader.hasName()) {
      String name = reader.getLocalName();
```

A questo punto è possibile chiamare il metodo *next()* con una certa sicurezza, in quanto tutti gli elementi XML hanno come figlio un nodo di testo che ne contiene il valore. Ad esempio, se reader punta all'elemento *<nome>*, sarà presente, come elemento successivo, un nodo testuale con il testo *"Mauro":*

```
reader.next();
```

A questo punto il programma estrae il valore dell'elemento tramite il metodo *getText()* e lo associa alle variabili dichiarate in precedenza. Per fare questo sono presenti una serie di blocchi *if*, uno per ciascuna variabile:

```
 if( "nome".equals(name) ) {
     nome = reader.getText();
     break;
 }
if( "cognome".equals(name) ) {
  cognome = reader.getText();
  break;
}
if( "via".equals(name) ) {
   via = reader.getText();
   break;
}
if( "numero".equals(name) ) {
   numero = reader.getText();
```

NOTA

UNA COMUNITÀ ATTIVA

Le API StAX sono state standardizzate dal JCP, l'organismo che si occupa di definire le tecnologie che entreranno a far parte degli standard ufficiali del linguaggio Java. Ogni singola specifica ha un numero progressivo. Quello di StAX è 173. Dalla relativa pagina sul sito di JCP (http://www.jcp.org/en/jsr/detail?id=173) emerge come l'approvazione finale sia datata 25 marzo 2004.

```
        break;
    }
    if( "cap".equals(name) ) {
        cap = reader.getText();
        break;
    }
    if( "localita".equals(name) ) {
        localita = reader.getText();
        break;
    }
    if( "provincia".equals(name) ) {
        provincia = reader.getText();
        break; }
    }
    break;
}
reader.next();
}
```

al termine delle operazioni i dati ottenuti vengono stampati a video:

```
System.out.println(nome + " " + cognome);
System.out.println(via + " " + numero);
System.out.println(cap + " " + localita + " (" +
                                        provincia + ")");
reader.close();
}
}
```

Un esempio di output è il seguente:

Destinatario:
====
Mauro Gallipari
Via Lombardia 1/2
20100 Milano (MI)

LETTURA DEGLI ATTRIBUTI

Come si nota osservando il documento XML prima riportato, alcuni elementi possiedono attributi.
Le API StAX forniscono numerosi metodi per manipolarli. Ad esempio, per ottenere il valore di un attributo è possibile utilizzare il metodo *getAttributeValue()* passando il numero progressivo di presenza dello stesso nel file XML. L'esempio seguente esegue una iterazione sul file degli ordini elaborando solo gli eventi di tipo *START_ELEMENT.* Nel caso gli elementi siano di tipo *<prodotto>*, viene estratto l'id e stampato a video. Si noti che gli indici di attributi iniziano da zero:

```
while (reader.hasNext()) {
  int type = reader.getEventType();
  if (type == XMLEvent.START_ELEMENT) {
    if (reader.hasName()) {
      String name = reader.getLocalName();
      if( "prodotto".equals(name) ) {
        String id = reader.getAttributeValue(0);
        System.out.println("id=" + id); } } }
  reader.next();
}
reader.close();
```

l'output prodotto è simile al seguente:
id=772666253
id=776747374
id=771826531
id=775425434

Non è possibile chiamare il metodo *getAttributeValue()* se l'evento non è di tipo *START_ELEMENT* oppure *ATTRIBUTE.* Per avere ulteriori informazioni sugli attributi è possibile utilizzare altri metodi. Il blocco di codice seguente elabora l'elemento XML *<ordine>* al fine di produrre l'elenco degli attributi con il relativo nome, tipo e valore. Per conoscere il numero di attributi dell'elemento corrente è possibile invocare il metodo *getAttributeCount():*

```
if( "ordine".equals(name) )
{
  System.out.println("elemento ordine");
  System.out.println("====");
  System.out.println("numero attributi = " +
                                reader.getAttributeCount());
  System.out.println("====");
  for (int i=0; i < reader.getAttributeCount(); i++)
  {
    System.out.println("nome            = " +
      reader.getAttributeLocalName(i));
    System.out.println("tipo = " +
                                reader.getAttributeType(i));
    System.out.println("valore = " +
                        reader.getAttributeValue(i) + "\n");
  }
}
```

L'output prodotto da questo programma è il seguente:

elemento ordine
====
numero attributi = 3
====
nome = id
tipo = CDATA
valore = 9938277378

nome = data
tipo = CDATA

NOTA

UN CAFFÈ MACCHIATO

Un sito interessante dove è possibile recuperare altre informazioni in merito alla tecnologia XML ed al suo utilizzo con Java è "Cafe con Leche" (http://www.cafeconleche.org/), che in spagnolo significa "caffè con latte".

valore = 20051201
nome = utente
tipo = CDATA
valore = AB01392

SCRIVERE DOCUMENTI

Come accennato in precedenza, è possibile utilizzare StAX anche per produrre documenti XML. Questa tipologia di file sono facilmente realizzabili anche scrivendo un file con le normali API Java, ad esempio attraverso la classe *FileOutputStream*. Il compito non è difficile: è sufficiente ricordarsi di iniziare il file con la dichiarazione *<?xml?>* e formattare il resto del contenuto in modo corretto. In questo modo, però, è necessario che il programmatore gestisca in prima persona e manualmente diverse cose, come i *namespace*.
StAX permette di generare documenti XML utilizzando API semplici e fornendo un supporto basilare alla gestione della struttura senza richiedere un particolare sforzo al programmatore.
Il programma seguente genera un semplice file XML che contiene solo l'elemento principale *<ordine>*. Questa volta, invece che ottenere una factory di input si ottiene una di output, attraverso la classe *XMLOutputFactory*. Da questa è possibile creare un oggetto *XMLStreamWriter* che dispone di una serie di metodi di scrittura. Questo oggetto è collegato ad uno di tipo *FileOutputStream* che si occupa della scrittura fisica dei dati su disco.
Nell'oggetto *XMLStreamWriter*, il metodo *writeStartDocument()* produce l'inizializzazione del file. Il metodo *writeStartElement()* scrive invece l'inizio di un elemento, mentre *writeAttribute()* scrive un attributo. Sono poi presenti i metodi *writeEndElement()* e *writeEndDocument()* per chiudere elemento e documento.
Al termine delle operazioni di scrittura è necessario chiudere l'oggetto *XMLStreamWriter* e il *FileOutputStream*, altrimenti i dati non saranno scritti effettivamente sul file.

```
package it.edmaster.ioprogrammo.stax;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

import javax.xml.stream.XMLOutputFactory;
import javax.xml.stream.XMLStreamException;
import javax.xml.stream.XMLStreamWriter;

public class CreaDocumento
{
  public static void main(String[] args) throws
                          XMLStreamException, IOException
  {
    XMLOutputFactory factory =
                    XMLOutputFactory.newInstance();

    FileOutputStream fos = new
                    FileOutputStream("ordine1.xml");
    XMLStreamWriter writer =
                    factory.createXMLStreamWriter( fos );

    //inizio del documento
    writer.writeStartDocument();

    //inizio elemento ordine
    writer.writeStartElement("ordine");

    //scrive attributi
    writer.writeAttribute("id", "991828298");
    writer.writeAttribute("data", "20051202");
    writer.writeAttribute("utente", "AB08273");

    //chiude elemento ordine
    writer.writeEndElement();

    //chiude documento
    writer.writeEndDocument();

    //chiude i file
    writer.close();
    fos.close();
  }

}
```

Il file prodotto da questo programma si chiama ordine1.xml e viene scritto nella directory corrente. Il contenuto è il seguente:

```
<?xml version="1.0" ?>
  <ordine id="991828298" data="20051202"
                              utente="AB08273">
  </ordine>
```

Per creare elementi innestati è sufficiente creare un nuovo elemento, attraverso il metodo *writeStartElement()* senza che quello precedente sia stato chiuso.

CONCLUSIONI

Le API StAX permettono di manipolare in modo abbastanza semplice i documenti XML, accedendo ad elementi ed attributi. Oltre al parsing delle informazioni presenti in documenti esistenti è possibile generare nuovi documenti XML utilizzando API tutto sommato di semplice utilizzo.

Massimiliano Bigatti

NOTA

A MODELLO

L'implementazione di riferimento di StAX, e cioè quella che SUN fornisce come modello di corretto funzionamento è disponibile sul sito di CodeHaus, all'indirizzo http://stax.codehaus.org/.

LINQ: Persistenza secondo Microsoft

Fra le novità più interessanti presentate per le prossime versioni di Visual Studio compare una tecnologia che risolve definitivamente il problema del mapping fra SQL e oggetti. Vediamo di cosa si tratta

.net

CD WEB
Linq.zip
cdrom.ioprogrammo.it

Nel mese di settembre 2005, a Los Angeles, si è tenuto uno degli appuntamenti più attesi dagli sviluppatori di tutto il mondo: la *Microsoft Professional Developer Conference* (PDC). L'appuntamento si tiene mediamente ogni due anni ed attrae migliaia di sviluppatori. Una delle tecnologie mostrate in anteprima ha suscitato molto clamore: LINQ. In questo articolo cercheremo di capire di cosa si tratta, quali vantaggi comporta. Parleremo della sua versione "pura" e delle sue due varianti: *DLinq* per la comunicazione con i Data Base e *XLinq* per la gestione dei files XML.

FONDAMENTI

Le metodologie di programmazione, nel corso degli anni, hanno subito svariati mutamenti generando, come effetto collaterale, una serie di linguaggi diversi. La maturazione di tali metodologie ha portato però ad uno "standard" ormai universalmente riconosciuto: l'*Object Oriented Programming* (OOP). Nel mondo OOP, ogni elemento del software viene concepito come un *Oggetto*, con chiaro riferimento al mondo reale. Tale rappresentazione però, sebbene estremamente vantaggiosa all'interno del codice, si scontra spesso con un altro tipo di rappresentazione: quella relazionale, inevitabile per la conservazione dei dati. Per meglio chiarire la problematica, la cosa migliore è fare un semplice esempio. Supponiamo che il nostro software debba gestire un catalogo di automobili. Secondo i principi della programmazione ad oggetti, la prima cosa che ci viene in mente è quella di creare un oggetto che possa rappresentare la nostra automobile:

```
using System;
namespace ioProgrammo.Tests
{
  public class Automobile
  {
     public string Marca;
     public string Modello;
     public string Colore;
     private bool IsNew;
     /// <summary>
     /// Costruttore di default
     /// </summary>
     public Automobile(int IDAutomobile){
        //recupera l'automobile specifica dal database
        IsNew = false;
     }
     public void Automobile(string Marca, string
                                    Modello, string Colore){
        //Crea un nuovo oggetto automobile che andrà
        //successivamente memorizzato nel database
        IsNew = true;
     }
     public void Avanti(){
     }
     public void Indietro(){
     }
     public void Frena(){
     }
     public void Salva(){
        //codice necessario per il salvataggio
        //dell'oggetto automobile nel database
     }
}
```

Il nostro oggetto *Automobile* ha due peculiarità individuabili nei rispettivi costruttori. Al primo infatti passiamo, ad esempio, un *IDAutomobile*. Tale valore verrà passato al codice che esegue l'interrogazione al database il cui scopo sarà quello di recuperare i dati corretti e di popolare i relativi campi del nostro

REQUISITI

Conoscenze richieste
.NET Framework, C#

Software
Visual Studio .net 2005 RTM, C# LINQ Tech Preview Update for Visual Studio 2005 RTM Release

Impegno

Tempo di realizzazione

COME INIZIARE

Microsoft mette a disposizione un'estensione per Visual Studio 2005 per effettuare i test. Si tratta della *"C# LINQ Tech Preview Update for Visual Studio 2005 RTM Release"* scaricabile da:
http://download.microsoft.com/download/4/7/0/4703eba2-78c4-4b09-8912-69f6c38d3a56/LINQ%20Preview.msi

oggetto. Il secondo costruttore invece accetta in ingresso i valori dei rispettivi campi. Una volta istanziato l'oggetto (ed eventualmente usato), potremmo richiamare il metodo *Salva* che inserirà i dati nella tabella del nostro database (operazione inversa a quella precedente). L'oggetto preso in esame è estremamente semplice, eppure il codice da scrivere per poter fare un mapping reale è parecchio. Dobbiamo infatti realizzare tutta la parte di comunicazione con il database, scrivere il codice necessario a generare le query, gestire eventuali errori ecc. Provate ad immaginare un software reale: esso conterrà molti più oggetti e sicuramente molto più complessi!! Torniamo al nostro piccolo oggetto. Oltre ai grossi vantaggi della programmazione ad oggetti (sicuramente non apprezzabili dal codice mostrato), chi usa degli editor evoluti per scrivere codice saprà apprezzare i vantaggi dell'intellisense. Si tratta di un meccanismo abbastanza complesso che permette di autocompletare il codice durante la digitazione. Nel nostro caso, presa l'istanza della classe *Automobile*:

```
Automobile auto = new Automobile(1);
auto.Colore = "Rosso";
```

nella seconda riga, dopo la digitazione del punto, appariranno tutte le proprietà ed i metodi pubblici dell'oggetto *Automobile* consentendo, da un lato una maggiore velocità nella scrittura del codice, e dall'altro la notifica di eventuali errori di digitazione già a design time. Interrogando direttamente un database invece, questi vantaggi si perdono, essendo quest'ultimo una entità "estranea" al nostro codice. Tutti questi problemi e queste necessità sentite dagli sviluppatori, hanno portato alla nascita di Linq e dei suoi derivati.

LINQ, DLINQ E XLINQ: CONCETTI DI BASE

Il nuovo *Language Integrated Query* è, come dice il termine stesso, un linguaggio di interrogazione "integrato". Per integrato si intende che, grazie a Linq, le query che scriveremo saranno comprensibili dal linguaggio .net in quanto saranno rappresentate da istruzioni note al compilatore. Che voi lavoriate in Visual Basic o in C# poco importa: sarà compito del CLR *(Common Language Runtime)* usarle in modo opportuno, convertendole in stringa se state interrogando un Data Base, producendo XML se state usando XLinq o usandole direttamente se state usando Linq.

Attenzione: quando parlo di Visual Basic e C# mi riferisco a Visual Basic 9.0 ed a C# 3.0, ovvero le prossime versioni di questi linguaggi!

Avere un linguaggio di interrogazione integrato significa in pratica poter scrivere questo:

```
var query = from c in custs, o in orders where
        o.Customer == c.Name select new {c.Name,
                        o.OrderID, o.Amount, c.Age };
```

Notate qualcosa di familiare? La query appena mostrata è la traduzione in Linq di

```
SELECT [t0].[Age], [t1].[Amount], [t0].[Name],
  [t1].[OrderID] FROM [Customers] AS [t0], [Orders]
      AS [t1] WHERE [t1].[Customer] = [t0].[Name]
```

Comodo Vero? Ma come è possibile tutto questo? E perché non lo abbiamo avuto prima? Il motivo è abbastanza semplice: nella precedente versione del framework mancavano tutta una serie di elementi indispensabili per la realizzazione di un sistema ottimizzato come Linq. Le funzionalità interne di Linq e dei suoi derivati infatti, sono state utilizzate usando nuove features come *Generics, Lambda Expressions* (evoluzione degli *Anonymous Method*), *Extension Methods* ecc. Abbiamo accennato a Linq, DLinq ed XLinq ma come si relazionano tra loro?
Come dice lo stesso nome, il primo è la base su cui poggiano gli altri due. Linq infatti contiene tutte le istruzioni generali che permettono la scrittura di query integrate: è praticamente il motore. La sua

OfType	Filter based on type affiliation
Select/SelectMany	Project based on transform function
Where	Filter based on predicate function
Count	Count based on optional predicate function
All/Any	Universal/Existential quantification based on predicate function
First/FirstOrDefault	Access initial member based on optional predicate function
ElementAt	Access member at specified position
Take/Skip	Access members before/after specified position
TakeWhile/SkipUntil	Access members before/after predicate function is satisfied
GroupBy	Partition based on key extraction function
ToDictionary	Create key/value dictionary based on key extraction function
OrderBy/ThenBy	Sort in ascending order based on key extraction function and optional comparison function
OrderByDescending/ ThenByDescending	Sort in descending order based on key extraction function and optional comparison function
Reverse	Reverse the order of a sequence
Fold	Aggregate value over multiple values based on aggregation function
Min/Max/Sum/Average	Numeric aggregation functions
Distinct	Filter duplicate members
Except	Filter elements that are members of specified set
Intersect	Filter elements that are not members of specified set
Union	Combine distinct members from two sets
Concat	Concatenate the values of two sequences
ToArray/ToList	Buffer results of query in array or List<T>
Range	Create a sequence of numbers in a range
Repeat	Create a sequence of multiple copies of a given value

Tabella 1: Sintesi dei comandi applicabili

.net

I TUOI APPUNTI

estensibilità ha permesso la creazione delle due varianti di cui parliamo in questo articolo. La possibilità di estendere Linq, lascia sperare che un domani possano esistere altre estensioni, magari per usare questa tecnologia anche su database diversi.

LINQ

Come abbiamo visto nei precedenti paragrafi, Linq è diviso in tre famiglie distinte. In questo paragrafo parleremo di Linq puro. Supponiamo di avere un Array di nomi e di voler selezionare tutti i nomi la cui lunghezza sia di 5 caratteri. Di seguito il codice che esegue questa operazione:

```
using System;
using System.Query;
using System.Collections.Generic;
class TestApp {
static void Main() {
string[] names = { "Michele", "Lucia", "Martina",
              "Lorenzo", "Catia", "Giuseppe",  "Nunzia"};
IEnumerable<string> expr = from s in names where
            s.Length == 5 orderby s select s.ToUpper();
foreach (string item in expr)
Console.WriteLine(item);}
}
```

A video vedremo apparire
Lucia
Catia

Che sono gli unici due nomi che soddisfano i criteri di filtro impostati nella query (*where s.Length == 5*). La stessa operazione avremmo potuto compierla anche oggi ma ci sarebbe costato molto più codice. Avremmo dovuto infatti ciclare manualmente all'interno di tutti gli elementi del nostro vettore, contare i caratteri di ognuno e mettere a video quelli corretti. È praticamente la stessa cosa che, internamente, fa Linq ma con il vantaggi di avere un sistema unico per eseguire la ricerca (la query) e, immagino, con prestazioni ottimizzate. Una delle cose più evidenti nel codice *TestApp* è la similitudine della stringa di interrogazione con le query che normalmente usiamo per interrogare i nostri Data Base. In Linq infatti, sono implementate la maggior parte delle istruzioni dell'SQL Standard (vedi **Tabella 1**).
Questo ci permette eventualmente di ordinare i dati, raggrupparli ecc. Tutte cose, come già detto, fattibili anche oggi ma con molto codice in più da scrivere.

DLINQ

Passiamo ora a vedere un esempio su DLinq, l'elemento forse più controverso di questa architettura.

Iniziamo con il creare due tabelle di esempio su Sql Server:

```
create table People (
        Name nvarchar(32) primary key not null,
        Age int not null, CanCode bit not null
)
```

per la tabella dei clienti e

```
create table Orders (
        OrderID nvarchar(32) primary key not null,
        Customer nvarchar(32) not null,
        Amount int
)
```

per la tabella degli ordini. Come abbiamo detto nella parte generale di questo articolo, grazie a DLinq abbiamo la possibilità di "mappare" le entità della base dati su oggetti del nostro codice. Nel caso specifico, dovremmo fare il modo che alla tabella *People* corrisponda un oggetto *People* e alla tabella *Orders* corrisponda il relativo oggetto.
Per la tabella *People* quindi procediamo in questo modo:

```
[Table(Name="People")]
public class Person {
  [Column(DbType="nvarchar(32) not null", Id=true)]
  public string Name;
  [Column]
  public int Age;
  [Column]
  public bool CanCode;
}
```

Allo stesso modo facciamo per la tabella *Orders*:

```
[Table(Name="Orders")]
public class Order
{
  [Column(DbType="nvarchar(32) not null", Id=true)]
  public string OrderID;
  [Column(DbType="nvarchar(32) not null")]
  public string Customer;
  [Column]
  public int? Amount;
}
```

Come è possibile notare dal codice, per mappare correttamente i nostri oggetti alle entità relazionali del Data Base, si fa uso degli attributi (vedi Box laterale). È grazie ad essi che il compilatore riesce a "capire" a cosa realmente corrispondono i campi del nostro oggetto. In tal modo sarà poi possibile sia caricare i dati dal database che memorizzarli. Uno degli aspetti più interessanti è quello che, i tipi di dati del database vengono effettivamente mappati con i

NOTA

Un buon punto di partenza per capire i concetti alla base dell'Object Oriented Programming è Wikipedia, più precisamente questo link: http://en.wikipedia.org/wiki/Object-oriented_programming. Partendo da esso, i più curiosi potranno approfondire tutti gli elementi che caratterizzano la programmazione OO. Per i più esigenti è invece consigliata la lettura di Thinking in Java, di Bruce Eckel: http://www.bruceeckel.com/

PER SAPERNE DI PIÙ

Per meglio comprendere cosa sono e come si usano gli attributi, fare riferimento a questo link su MSDN: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/csref/html/vclrfintroductiontoattributes.asp

tipi di dati .net. Uno dei problemi maggiori riscontrati con le versioni precedenti del .net framework era, in particolar modo, quello dei tipi nulli, presenti nei DataBase ma non nel framework. Quelle che abbiamo appena creato sono due classi che devono essere istanziate con i dati recuperati dalla base dati. Vediamo quindi come procedere:

```
// stabiliamo un query context via ADO.NET sul
//Data Base
DataContext context = new DataContext("Initial
    Catalog=TestDataBase;Integrated Security=sspi");
// recuperiamo le variabili che corrispondono ai nostri
//oggetti Person e Order che abbiamo mappato sul
//Data Base
Table<Person> custs = context.GetTable<Person>();
Table<Order> orders   = context.GetTable<Order>();
// costruiamo la query
var query = from c in custs, o in orders where
            o.Customer == c.Name select new {c.Name,
                        o.OrderID, o.Amount, c.Age};
// eseguiamo la query
foreach (var item in query)
Console.WriteLine("{0} {1} {2} {3}", item.Name,
                item.OrderID, item.Amount, item.Age);
```

Come prima cosa, creiamo il nostro "contesto" *(context)* in cui la query deve essere eseguita. È questo il punto in cui specifichiamo qual è il Data Base da utilizzare e come accedervi.
Lo step successivo sarà quello di collegare le tabelle alle nostre classi:

```
Table<Person> custs = context.GetTable<Person>();
Table<Order> orders   = context.GetTable<Order>();
```

solo a questo punto possiamo creare ed eseguire la nostra query. Internamente essa sarà trasformata in una classica query SQL identica a quella che scriveremmo oggi:

```
SELECT [t0].[Age], [t1].[Amount], [t0].[Name],
   [t1].[OrderID] FROM [Customers] AS [t0], [Orders]
        AS [t1] WHERE [t1].[Customer] = [t0].[Name]
```

I Data Base (SQL Server nel nostro caso) infatti, necessitano di una semplice stringa in ingresso che verrà processata dal loro motore. A questo punto ci sono due cose interessanti da notare. La prima è che, in fase di esecuzione, il mapping reale non viene eseguito finché la query non viene processata (nel codice riportato avviene nel ciclo *foreach*) e questo è importante ai fini del risparmio di memoria. La seconda cosa importante è che la query è scritta direttamente in C# quindi, eventuali errori sintattici che a runtime provocherebbero un malfunzionamento del software, emergono già in fase di compilazione.

XLINQ

Diamo ora un rapido sguardo all'ultimo elemento della famiglia Linq: XLinq. Grazie ad esso possiamo operare in xml con la stessa semplicità che abbiamo riscontrato negli esempi precedenti. XLinq, oltre a darci la possibilità di interrogare l'XML, ci fornisce anche un ottimo sistema per la sua rappresentazione in memoria, che si rivela estremamente comodo in parecchi scenari.
Facciamo subito qualche esempio.

```
var e = new XElement("Person",
new XAttribute("CanCode", true),
new XElement("Name", "Michele Locuratolo"),
new XElement("Age", 29));
var s = e.ToString();
```

Il codice sopra riportato è esattamente la rappresentazione del seguente codice XML:

```
<Person CanCode="true">
<Name> Michele Locuratolo </Name>
<Age>29</Age>
</Person>
```

Vediamo ora un secondo esempio che ci permette di capire quanto sia comodo XLinq. Supponiamo di voler trasformare in XML il risultato di una query (ad esempio la nostra collection di oggetti mappata al Data Base dell'esempio su DLinq). Tutto quello che dobbiamo fare è semplicemente:

```
var x = new XElement("People", from p in people
                  where p.CanCode select new XElement(
    "Person", new XAttribute("Age", p.Age), p.Name));
```

Come visto nell'esempio precedente, *XElement* sarà la rappresentazione in memoria di un XML che, nel nostro caso potrebbe essere:

```
<People>
   <Person Age="3">Martina Locuratolo</Person>
   <Person Age="29">Michele Locuratolo</Person>
</People>
```

ALCUNE CONSIDERAZIONI

Linq è una tecnologia appena nata Dare un parere ben definito oggi sarebbe un errore. Per quello che ho potuto vedere, Linq e XLinq sono due tecnologie molto interessanti e davvero comode.
Come avete visto negli esempi, semplificheranno molto rispettivamente la gestione delle collections che dei files xml.
Rimane qualche perplessità sulle performance, ma staremo a vedere cosa accadrà...

Michele Locuratolo

.net

APPROFONDIMENTI

Il punto di partenza per chi vuole approfondire Linq è http://msdn.microsoft.com/netframework/future/linq/.
Molto consigliata la lettura di questo articolo http://msdn.microsoft.com/netframework/future/linq/default.aspx?pull=/library/en-us/dndotnet/html/linqprojectovw.asp di Don Box da cui sono tratti anche gli esempi citati qui.

CONTATTA L'AUTORE

L'autore può essere contattato attraverso il suo blog su http://blogs.mindbox.it e sarà lieto di rispondere alle domande dei lettori.

Estendere Visual Studio .NET

Il modello ad oggetti di Visual Studio permette di estendere l'ambiente di sviluppo in ogni suo aspetto e di aggiungervi nuove funzionalità. Vediamo come personalizzare il nostro IDE preferito

C#

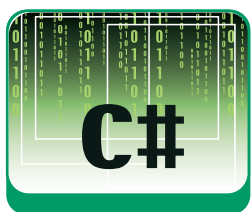
CD WEB
Linecomputer.zip
cdrom.ioprogrammo.it

REQUISITI

Conoscenze richieste
conoscenze medio-elevate di C#

Software
.NET framework SDK 1.1, Visual Studio .NET 2003, Windows 2000 o successivi

Impegno

Tempo di realizzazione

Per creare nuovi tool che aiutino lo sviluppo delle applicazioni in Visual Studio .NET, l'ambiente di sviluppo della Microsoft mette a disposizione un modello di automazione, conosciuto nelle vecchie versioni anche come extensibility model che, altro non è che una serie di interfacce di programmazione, attraverso le quali è possibile interagire con i meccanismi interni che regolano ogni singola funzionalità dell'IDE stesso. Se pensate ad ogni singola operazione che Visual Studio .NET vi permette di fare, dalla creazione di soluzioni e progetti, all'aggiunta di file o classi, alla scrittura di codice, fino alla compilazione del codice, essa può essere svolta in maniera programmatica, grazie appunto al suo automation model. In questo articolo utilizzeremo il modello di automazione per realizzare un add-in che data una soluzione di Visual Studio, conti le linee di codice che la costituiscono, suddividendole sia per file, sia per classe, in modo da avere semplici metriche sui nostri progetti, ad esempio per valutare quali classi sono quelle più consistenti, e che magari andrebbero reingegnerizzate, spezzate in più classi e snellite.

LA CLASSE CONNECT

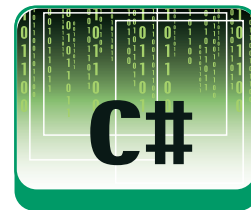
Una volta terminata la creazione della soluzione, noterete che essa contiene due progetti, il primo è quello dell'*Add-in* vero e proprio, mentre il secondo è il progetto di setup con cui potrete distribuirlo ed installarlo. Il primo progetto contiene il file *Connect.cs*, in cui è stato creato uno scheletro della classe *Connect*, che costituisce l'oggetto principale di ogni Add-in, ed il punto di partenza della sua esecuzione. Essa è inoltre farcita di commenti che vi consiglio di leggere per comprenderne meglio il suo funzionamento. La classe *Connect* implementa le interfacce *IDTExtensibility2* e *IDTCommandTarget*, e quindi conterrà già una serie di metodi che toccherà a noi personalizzare o implementare da zero. Il metodo *OnConnection* viene eseguito alla notifica che l'Add -in è stato caricato nell'ambiente, esso riceve in ingresso come primo parametro un oggetto *application* di classe *_DTE* (DTE sta per *Development Tools Extensibility*) che rappresenta l'IDE stesso, e attraverso il quale è possibile ricavare tutti gli oggetti che Visual Studio mette a disposizione. Il secondo parametro *connectMode* può assumere 5 diversi valori, che rappresentano la modalità con cui l'add-in è stato caricato nell'ambiente:

- **ext_cm_AfterStartup:** l'add-in è stato caricato dopo che l'applicazione stessa è stata avviata, in genere selezionandolo dall'*Add-in Manager.*
- **ext_cm_Startup:** l'add-in è stato caricato contestualmente allo *startip* dell'IDE.
- **ext_cm_External:** l'add-in è stato caricato da un'altra applicazione o da un componente esterno all'IDE.
- **ext_cm_CommandLine:** l'add-in è stato caricato da linea di comando, attraverso i parametri del comando devenv (è l'eseguibile di Visual Studio).
- **ext_cm_Solution:** L'add-in è stato caricato da una soluzione che lo richiede.
- **ext_cm_UISetup:** l'add-in è stato caricato per la prima volta dopo la sua installazione.

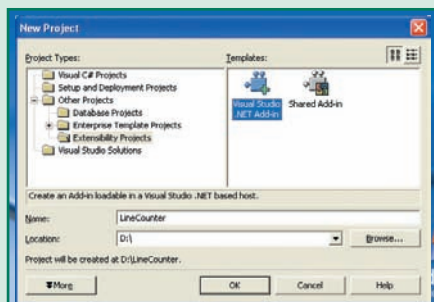
Il terzo parametro del metodo, addInInst, è l'oggetto che rappresenta l'istanza dell'add-in. Tale istanza è passata al metodo *AddNamedCommand* dell'oggetto *Commands* per aggiungere una voce al menu *Tools*, con la quale avviare il comando . Il metodo *AddNamedCommand* restituisce un oggetto *Command*, che aggiungiamo fra i campi privati della classe *Connect*, in modo da averlo a disposizione in tutti i suoi metodi.

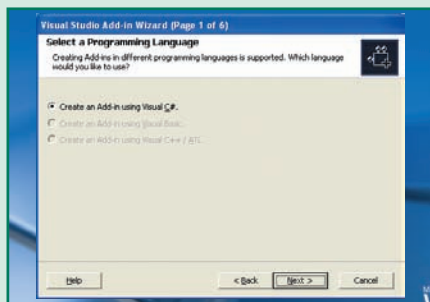
```
private Command command;
```

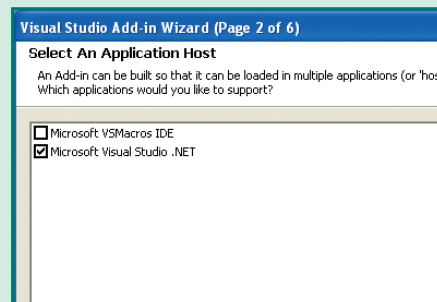
Nel nostro caso il metodo *OnConnection* sarà dunque fatto così:

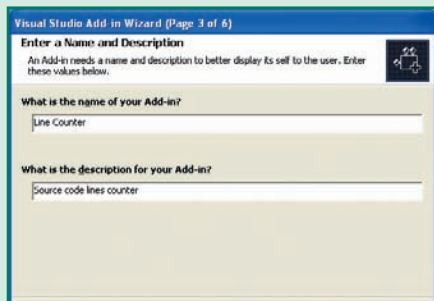
```
public void OnConnection(object application,
    Extensibility.ext_ConnectMode connectMode, object
                   addInInst, ref System.Array custom)
  {
      applicationObject = (_DTE)application;
      addInInstance = (AddIn)addInInst;
      if(connectMode == Extensibility.ext_
                   ConnectMode.ext_cm_UISetup
        || connectMode == Extensibility.ext_
                   ConnectMode.ext_cm_AfterStartup)
        {  object []contextGUIDS = new object[] { };
          Commands commands =
                   applicationObject.Commands;
          _CommandBars commandBars =
                   applicationObject.CommandBars;
          try
          { command = commands.AddNamedCommand(
            addInInstance, "LineCounter", "Line Counter",
              "Executes 'Line Counter' for the current
                project", true, 59, ref contextGUIDS,
                   (int)vsCommandStatus.
                 vsCommandStatusSupported+(int)
            vsCommandStatus.vsCommandStatusEnabled);
            CommandBar commandBar =
                 (CommandBar)commandBars["Tools"];
            CommandBarControl commandBarControl =
              command.AddControl(commandBar, 1);}
          catch(System.Exception /*e*/)
          { //esiste già il comando?}
      }
  }
```

COME INIZIARE

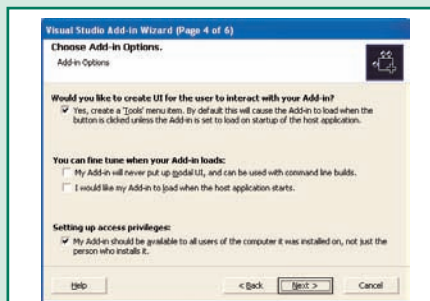
Visual Studio .NET mette a disposizione un wizard apposito per la creazione degli Add-in. Ecco i passi necessari per iniziare da zero l'implementazione di un nuovo Add-in.

> UN NUOVO PROGETTO

1 Dal menù file, selezionare la voce nuovo progetto, quindi dalla categoria Extensibility Project scegliere la tipologia Visual Studio .NET Addin e premere OK.

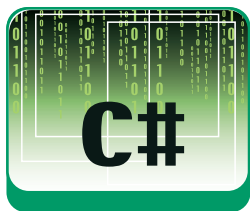
> IL LINGUAGGIO

2 È necessario scegliere quale linguaggio si vuole utilizzare per scrivere l'Add-in, le scelte possibili dipendono dalla propria installazione, scegliamoVisual C#.

> CHI É L'HOST?

3 Il terzo passo riguarda la scelta dell'applicazione host dell'Add-in. Selezionate Visual Studio .NET e premete sul pulsante avanti.

> UN NOME PER L'ADD IN

4 Assegniamo un nome, ad esempio Line Counter ed una descrizione all'Add-in. La descrizione apparirà nella finestra Add-in Manager di Visual Studio.

> QUALE MENU

5 Selezionate la prima Checkbox, per aggiungere una voce al menù Tools da, e la quarta per permetterne l'utilizzo a tutti gli utenti della macchina.

> FINITO

6 A questo punto sono state raccolte tutte le informazioni necessarie per iniziare la creazione dell'Add-in, non resta che premere il pulsante Finish.

Notate che viene verificato che il *connectMode* abbia un valore fra *ext_cm_UISetup* oppure *ext_cm_AfterStartup*, in maniera che il comando venga aggiunto al menu *Tools*, sia alla prima esecuzione dopo il setup, oppure direttamente mediante il caricamento manuale dall'*Add-in manager.*
Il metodo *OnDisconnection* è il complementare del precedente, viene invocato quindi quando l'add-in viene scaricato dall'IDE, ad esempio alla chiusura dell'ambiente. Ad esempio in tale metodo è possibile eliminare il comando dal menu *Tools:*

```
public void OnDisconnection(Extensibility.ext_
                        DisconnectMode disconnectMode,
                           ref System.Array custom)
{ if(command!=null)
    command.Delete();
}
```

OUTPUT IN VISUALSTUDIO

Visual Studio .NET visualizza il testo in uscita da un'applicazione, o altro testo informativo, ad esempio risultati di compilazione, errori, e così via, in dei pannelli di output, chiamati OutputWindowPane. L'add-in di esempio utilizza un pannello di output personalizzato. Il metodo OutputWriteLine della classe SolutionCodeCounter mostra come realizzare un pannello del genere.

Quando l'utente clicca sul comando del menu *Tools*, per avviare l'add-in, viene generato un evento *QueryStatus*, e quindi invocato il metodo omonimo *QueryStatus*, in cui viene verificato quale comando è stato avviato, e quale sia il suo stato, ad esempio se è in quel momento abilitato o meno.
Dopo l'evento *QueryStatus*, se il comando è abilitato e dispobibile, viene eseguito il metodo *Exec*.
Questo è il metodo in cui inserire il codice che vogliamo far eseguire all'add-in. Nel nostro caso verifichiamo che vi sia una soluzione aperta, e se la risposta è affermativa, creremo un oggetto *SolutionCodeCounter*, mediante il quale ricaveremo le linee di codice della nostra soluzione e le mostreremo attraverso l'interfaccia utente del nostro add-in, cioè una dialog *LineCounterForm*.

```
public void Exec(string commandName,
   EnvDTE.vsCommandExecOption executeOption, ref
     object varIn, ref object varOut, ref bool handled)
{ handled = false;
   if(executeOption == EnvDTE.vsCommandExecOption
                    .vsCommandExecOptionDoDefault)
   {  if(commandName ==
                    "LineCounter.Connect.LineCounter")
      { if (applicationObject.Solution.IsOpen == true )
         { SolutionCodeCounter codeCounter = new
              SolutionCodeCounter(applicationObject);
           codeCounter.Count();
           LineCounterForm form=new LineCounterForm(
                 codeCounter,this.applicationObject);
           form.ShowDialog(); }
         else
         { System.Windows.Forms.MessageBox.Show(
              "Apri un progetto C# prima di lanciare il
                                   LineCounter!");}
         handled = true;
      return; }
   }
}
```

Nel caso in cui non vi sia alcuna soluzione aperta nell'IDE, mostriamo una *MessageBox* con un messaggio di errore. Prestate attenzione al fatto che il progetto dell'add-in non contiene alcun riferimento all'assembly *System.Windows.Forms.dll,* quindi è necessario aggiungerlo per utilizzare le sue classi dall'add-in.

DEBUG E TEST DELL'ADD-IN

Prima di andare avanti nell'implementazione delle classi necessarie per le funzionalità dell'add-in, vediamo come eseguirlo, testarlo e debuggarlo. Se lanciate il progetto in modalità di Debug, noterete che verrà lanciata una seconda istanza di Visual Studio .NET, ed alla prima esecuzione verrà eseguito il metodo *OnConnection* per l'installazione dell'Add-In. Se l'add-in non viene eseguito automaticamente, aprite l'*add-in manager* di Visual Studio dal menu *Tools,* e selezionate *Line Counter* da quelli disponibili, come in **Figura 1**.

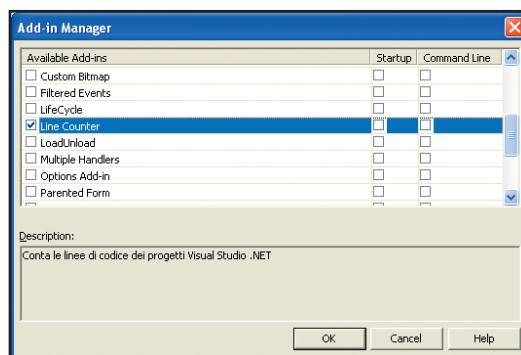



Fig. 1: ***L'add-in manager di Visual Studio***

Ciò avrà come effetto l'aggiunta di una voce *"Line Counter"* allo stesso menu *Tools*, mediante il quale lanciare l'add-in, come già detto. È possibile naturalmente introdurre breakpoint nell'istanza originale di Visual Studio, cioè in quella contentente il codice dell'add-in, e quindi effettuarne il debug, e seguirne l'esecuzione passo passo. Inoltre è possibile sfruttare l'output stesso di Visual Studio, che avviene sull'apposita finestra richiamabile. Se non

fosse già visualizzata, dal menu *View->Other Windows->Output*. È possibile aggiungere a tale finestra una categoria personalizzata di output, aggiungendo un nuovo *OutputWindowPane* alla collezione predefinita. Il metodo seguente ad esempio crea, se già non esiste, una finestra di output *"LineCounter"*, e vi stampa una linea di testo:

```
private void OutputWriteLine(string str)
{ Window win = this.devEnv.Windows.Item(
                        Constants.vsWindowKindOutput);
  OutputWindow window = (OutputWindow) win.Object;
  OutputWindowPane outPane=null;
  foreach(OutputWindowPane pane in
                        window.OutputWindowPanes)
  { if(pane.Name=="LineCounter")
     { outPane=pane;
        break;
     }
  }
  if(outPane==null)
     outPane=window.OutputWindowPanes.Add(
                                "LineCounter");
     try
     { outPane.OutputString(str+"\n"); }
  catch
  {/*ignore*/}
}
```

COMINCIAMO A CONTARE

Nel metodo *Exec* della classe *Connect* viene creata l'istanza di una classe *SolutionCodeCounter*, che è quella che si occupa di esplorare il codice della soluzione aperta nell'IDE, e raccogliere le informazioni riguardanti in questo caso il numero di linee del sorgente. Abbiamo bisogno di conoscere l'istanza dell'IDE contenente la soluzione, quindi avremo un campo privato di tipo *_DTE*, che inizializzeremo nel costruttore:

```
private _DTE devEnv;
private Hashtable fileProjects;
public SolutionCodeCounter(_DTE app)
{ devEnv=app;
   fileProjects=new Hashtable();
}
```

Nll'hashtable *fileProjects* invece inseriremo tutti le informazioni sui file sorgenti che costituiscono la soluzione e che troveremo esplorandola. Per ricavare tali informazioni dovremo utilizzare parecchi degli oggetti che il modello di automazione di Visual Studio ci mette a disposizione.
Il metodo principale della classe *SolutionCodeCounter* è il metodo *Count*, dentro al quale viene avviata l'esplorazione della soluzione.

```
public void Count()
{  try
   { ExploreSolution(); }
    catch ( Exception ex )
    { this.devEnv.StatusBar.Text = ex.ToString();
       System.Windows.Forms.MessageBox.Show(
                                ex.ToString() );
    }
}
private void ExploreSolution()
{ EnvDTE.Solution sln  = devEnv.Solution;
   ExploreProjects( sln );
}
```

Il metodo *ExploreSolution* si occupa di ricavare l'istanza *Solution* che rappresenta la soluzione aperta e quindi invoca il metodo *ExploreProjects*, dentro al quale iteriamo sui vari progetti che essa può contenere per mezzo della collezione *Projects* della classe *Solution*.

```
private void ExploreProjects(Solution sln)
{ Project prj;
   //parte da 1
   for(int i=1,projectCount=sln.Projects.Count;i<=
                                projectCount;i++)
   { prj=sln.Projects.Item(i);
      if(prj.FullName.EndsWith(".csproj"))
      { try
         {  ExploreProjectFiles(prj);
             ExploreProjectCodeModel(prj.CodeModel);
         }
         catch(Exception ex)
         { OutputWriteLine(ex.Message); }
```

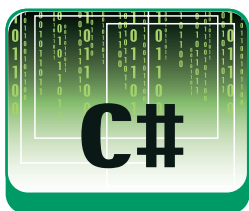
Fig. 2: *L'automation object model di Visual Studio*

C#

NOTA

IL PROGETTO

Sul CD o sul sito web di IoProgrammo trovate il codice completo dell'applicazione. Esso è costituito da una soluzione per Visual Studio 2003, che contiene il progetto dell'add-in ed il relativo progetto di setup per la sua installazione.

```
        }
    }
}
```

Come prima azione verifichiamo che il progetto abbia estensione *.csproj*, naturalmente se siete interessati agli altri tipi di progetti, ad esempio Visual Basic.NET, potete tranquillamente apportare qualche semplice modifica al metodo.
Per ogni progetto iniziamo quindi a ricaverne i file che lo costituiscono prima, nel metodo *ExploreProjectFiles*, e poi invece ne ricaviamo anche i diversi tipi in essi implementati.

FILE E TIPI

Per mantenere in memoria le informazioni trovate sui diversi file e contarne le linee, utilizziamo una classe *FileData* fatta così:

```
public class FileData
{
    private ArrayList types;
    private string name;
    private int lines;
    public FileData(string name)
    {
        types=new ArrayList();
        lines=0;
        this.name=name;
        CountLines();
    }
    private void CountLines()
    {
        StreamReader r=File.OpenText(name);
        string line;
        while( (line=r.ReadLine())!=null)
        {
            lines++;
        } }
    //...
}
```

Il metodo seguente invece, a partire da ogni istanza *Project*, ne ricava gli elementi che costituiscono il progetto, oggetti di classe *ProjectItem*, e che saranno file e directory. Quindi verifica in questo caso che il loro nome abbia estensione *.cs*, ed in caso affermativo crea un oggetto *FileData* e lo aggiunge alla hashtable *projectFiles:*

```
private void ExploreProjectFiles(Project prj)
{
    ProjectItem projectFile;
    FileData file;
    string path=Path.GetFullPath(prj.FullName);
    string dir=prj.FullName.Substring(
                        0,prj.FullName.LastIndexOf(
                        Path.DirectorySeparatorChar));
    for(int i=1,files=prj.ProjectItems.Count;i<=files;i++)
    {
        projectFile=prj.ProjectItems.Item(i);
        if(projectFile.Name.EndsWith(".cs"))
        {
            file=new FileData(Path.Combine(
                                dir,projectFile.Name));
            projectsFiles.Add(file.FullName,file);
            OutputWriteLine(String.Format("{0}: {1}
                        lines",file.Name,file.Lines));
        }
    }
}
```

Analogamente è possibile esplorare il *CodeModel* di ogni progetto, ricavandone ad esempio le classi o gli altri tipi, ed andando ancora più a fondo, metodi, variabili e così via. In questo caso le informazioni ricavate vengono conservate in un oggetto *TypeData*, il cui scheletro è il seguente:

```
public class TypeData
{
    private CodeElement element;
    private int startLine;
    private EnvDTE.vsCMElement kind;
    public string name;
    public string fullPath;
    public int lines;
}
```

I campi *kind* ed *element*, permettono di conoscere il tipo di elemento contenuto, ad esempio *CodeClass, CodeInterface*, e così via, e di ottenere informazioni sull'elemento stesso, ad esempio da quale linea esso inizia e dove termina. L'add-in completo, di cui trovate il codice allegato alla rivista, permette di ottenere sia il numero di linee dei file della soluzione, sia il numero di linee di ogni classe, interfaccia, struct o enum. La **Figura 3** mostra l'interfaccia

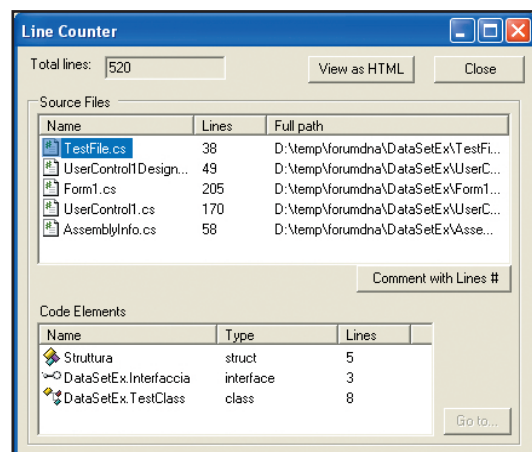



Fig. 3: *L'interfaccia dell'Add-in in esecuzione*

BIBLIOGRAFIA

• **INSIDE VISUAL STUDIO .NET 2003**
Johnson, Skibo, Young
(Microsoft Press)

•**MASTERING VISUAL STUDIO .NET - 2003**
(O'Reilly)

• **APPLIED MICROSOFT .NET PROGRAMMING**
Richter
(Microsoft Press)

dell'Add-in in esecuzione. I risultati così ricavati, possono essere anche salvati su un file, in maniera da consultarli anche in un seconto tempo. L'add-in permette di generare un file html e salvarlo in una sottocartella all'interno di quella contenente la soluzione ed aprire tale file nel browser integrato in VisualStudio. Ciò è possibile eseguendo il comando *View.URL* e passando come parametro il nome del file da aprire:

```
GenerateHTMLFile( codeCounter, sName,
                        htmlFileName, saveDirectory );
// Open html Page in VS .NET Browser
applicationObject.ExecuteCommand("View.URL",
                                          htmlFileName);
```

La **Figura 4** mostra Visual Studio .NET, dopo aver aperto nel suo browser interno il file HTML generato per la soluzione corrente. Notate anche in basso il pannello di output personalizzato per l'add-in *LineCounter.*

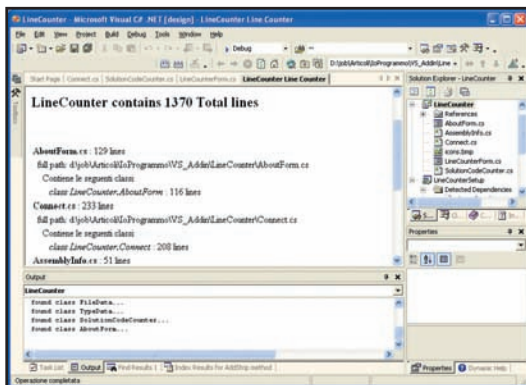



Fig. 4: File html generato nel browser di visual studio

INTERAGIRE CON L'IDE

L'add-in *LineCounter,* a partire da ogni file, mostra anche i tipi che in esso sono implementati, ed è possibile, ancora attraverso il modello di automazione, aprire un determinato file, o addirittura posizionarso proprio su una linea particolare, ad esempio sull'intestazione di una classe, o selezionare un singolo metodo. Infatti dalla collezione dei *TypeData* ricavata grazie alla classe *SolutionCodeCounter,* possiamo ricavare un particolare *CodeElement* e ad esempio ricavarne il file in cui è implementato e la linea esatta della sua dichiarazione. La **Figura 5** mostra l'add-in in primo piano e sullo sfondo Visual Studio che mostra l'intestazione del tipo selezionato attraverso il pulsante *GoTo.* A partire dal *CodeElement* selezionato, ne ricaviamo il *ProjectItem* che lo contiene e lo apriamo nell'IDE per mezzo del metodo Open.
A questo punto possiamo anche ottenere il punto di partenza dell'elemento stesso, grazie al metodo *GetStartPoint,* e di fine con *GetEndPoint,* e posizionarci sull'esatto punto del file che contiene l'elemento di codice:

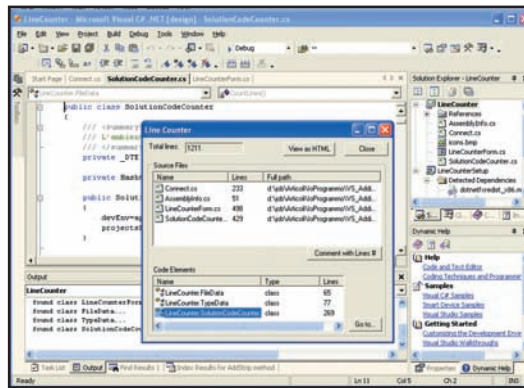



Fig. 5: Posizionamento su un elemento della soluzione

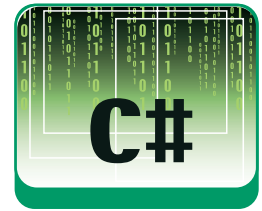
```
CodeElement elem=td.TheType;
Window win = elem.ProjectItem.Open(
                            Constants.vsViewKindCode);
if (win != null)
{  win.Activate();
   TextPoint start = elem.GetStartPoint(
                         EnvDTE.vsCMPart.vsCMPartWhole);
   TextPoint end = elem.GetEndPoint(
                         EnvDTE.vsCMPart.vsCMPartWhole);
   TextSelection txtSelect = start.Parent.Selection;
   txtSelect.MoveToPoint(start, false);
   //txtSelect.MoveToPoint(end, true);
   try
   { start.TryToShow(EnvDTE.vsPaneShowHow
                              .vsPaneShowTop, end);
    }
    catch(Exception ex)
   { applicationObject.StatusBar.Text=ex.Message;}
}
```

Questi sono semplici esempi di interazioni con l'ambiente di Visual Studio. Sottolineiamo solo il fatto che è possibile effettuare e personalizzare qualunque operazione messa a disposizione dall'interfaccia utente e non.

CONCLUSIONI

L'add-in realizzato, sebbene un semplice esempio creato per questo articolo, ha anche una certa utilità pratica, è servito a mostrare i passi necessari a realizzare un Add-in per Visual Studio .NET, come testarlo e debuggarlo, e come interagire con i meccani smi interni dell'IDE Microsoft. A partire da ciò, ogni lettore interessato può pensare e progettare le proprie estensioni per l'ambiente di sviluppo, e magari diffonderlo nella nostra comunità di noi sviluppatori Windows.

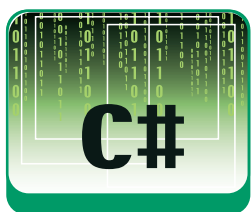
ing. Antonio Pelleriti

C#

CONTATTA L'AUTORE

Potete contattare l'autore per suggerimenti, critiche o chiarimenti all'indirizzo e-mail antonio.pelleriti@ioprogrammo.it**, oppure attraverso il sito** www.dotnetarchitects.it

Uno screenshot con Pinvoke!

Per realizzare un'applicazione . NET che catturi un'istantanea del desktop è necessario interagire con le API native di Windows, vedremo come farl con la tecnologia Platform Invoke

CD WEB
.visualstudio_net_addin.zip
cdrom.ioprogrammo.it

PInvoke è la tecnologia che permette di invocare funzioni unmanaged, scritte cioè in codice nativo e compilate in una DLL nativa, da un progetto in codice managed, cioè scritto in un linguaggio del .NET Framework. L'utilizzo classico dei servizi di Platform Invocation è quello di chiamare le API del sistema operativo Windows, contenute nelle DLL di sistema, e per le quali non esiste una implementazione managed. Platform Invoke, come del resto tutto il mondo .NET, basa le proprie caratteristiche sul concetto fondamentale di metadati, grazie al quale riesce ad identificare le funzioni native ed effettuare le necessarie traduzioni di tipi fra i due mondi.

UTILIZZARE UNA FUNZIONE NATIVA

Le DLL di sistema di Windows, contenute in genere nella directory %WINDIR%\System32, contengono molte funzioni che possono essere utili in un'applicazione .NET. Il modo di impiegarle in .NET dipende dal linguaggio del framework che vogliamo utilizzare. In C# è necessario l'utilizzo delle *keyword* static ed *extern*, per dichiarare che il metodo che l'implementazione del metodo è contenuta in una locazione esterna al codice .NET stesso, mentre tramite l'attributo *DllImport* si definisce quale DLL contiene la funzione nativa da invocare.

Una funzione nativa è identificabile tramite la propria signature, cioè firma, che è composta dal suo nome, dal numero e tipo di argomenti in ingresso, e dal tipo di ritorno. Vediamo un semplice esempio prima di passare a qualcosa di più complesso. Supponiamo di voler utilizzare la funzione *Beep* che emette il classico suono di sistema. Essa è esposta dalla dll *kernel32.dll,* con questa firma:

```
int Beep(DWORD frequency,DWORD duration);
```

per utilizzare tale funzione in .NET creeremo un metodo senza definirne il corpo, in questa maniera:

```
[DllImport("kernel32.dll")]
public static extern int Beep(int frequency, int
                                        duration);
```

Come potete notare, i parametri *DWORD* sono stati sostituiti con due interi. Vedremo che sarà fondamentale imparare a tradurre nel modo corretto i tipi nativi, altrimenti potrebbero verificarsi malfunzionamenti molto difficili da scovare, perché in molti casi non verrebbe generata alcuna eccezione, dato che le API Win32 utilizzano il valore di ritorno per segnalare una situazione di errore. La **Tabella 1** contiene un riferimento rapido per tradurre i tipi predefiniti, come interi, caratteri e stringhe.

L'ATTRIBUTO DLLIMPORT

Nell'esempio precedente abbiamo utilizzato l'attributo DllImport per marcare un metodo la cui implementazione si trova in una dll nativa. È possibile anche dare un nuovo nome alla funzione, utilizzando l'attributo *DllImport* con il campo *EntryPoint,* per definire quale funzione della DLL si vuole invocare e quindi dando un nome a nostro piacimento al metodo managed. È buona norma di programmazione creare delle classi contenente i metodi importati da una dll nativa, ad esempio:

```
class APIKernel32
{
   [DllImport("kernel32.dll",EntryPoint="Beep")]
   public static extern int Suono(int frequency, int
                                        duration);
}
```

A questo punto per invocare il metodo da una qual-

REQUISITI

Conoscenze richieste
conoscenze medie di C# e delle API Windows

Software
.NET Framework 1.1, Windows 2000/XP

Impegno

Tempo di realizzazione

siasi parte della nostra applicazione, basterà scrivere::

```
APIKernel32.Suono(500,1000);
```

Una dll nativa può contenere anche altri elementi oltre alle funzioni ad esempio costanti da utilizzare come argomento, o strutture. Prendiamo ad esempio la funzione *MessageBeep* contenuta in *user32.dll* e che permette di emettere uno dei classici suoni di sistema. I possibili valori del suo unico parametro sono delle costanti, che possiamo raggruppare in un tipo enumerativo

```
public enum MessageBeepType
{
   MB_ICONHAND = 0x00000010,
   MB_ICONQUESTION = 0x00000020,
   MB_ICONEXCLAMATION = 0x00000030,
   MB_ICONASTERISK = 0x00000040
}
```

quindi la definizione del metodo saràla seguente:

```
[DllImport("user32.dll")]
public static extern int MessageBeep(
                              MessageBeepType uType);
```

La **Figura 1** mostra un'applicazione di esempio (vedi CD allegato), che utilizza le funzioni *Beep* e *MessageBeep*, come visto fin'ora.

Beep frequency 1000 duration 1000
Message Beep
Play Asterisk Hand
Question Exclamation

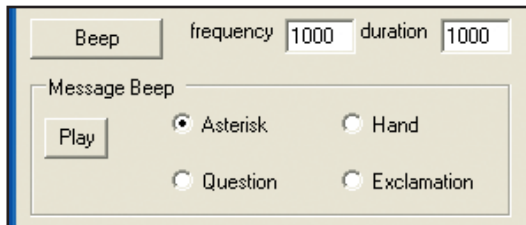



Fig. 1: **Emettere suoni di sistema con PInvoke**

CODICI DI ERRORE

Le API Win32 forniscono la funzione *GetLastError* per ricavare il codice dell'ultimo errore che si è verificato. È però sconsigliato utilizzare tale funzione mediante i servizi di platform invoke, perché i risultati potrebbero essere falsati a causa dell'interazione fra CLR e sistema operativo, è bene invece utilizzare l'attributo *DllImport* impostando a *true* il campo *SetLastError*. In tale maniera si indica che la funzione nativa chiamata imposti il codice di errore, che poi sarà ricavabile mediante il metodo managed *Marshal.GetLastWin32Error*. Supponiamo ad esempio di voler ricavare il testo di una finestra visualizzata sul nostro desktop, di cui conosciamo o possiamo ricavare l'handle, utilizzando la funzione *GetWindowText* di *user32.dll*. Scriviamo quindi le due dichiarazioni seguenti in C#:

```
[DllImport("user32.dll", SetLastError=true, CharSet=
                                        CharSet.Auto)]
public static extern int GetWindowText(IntPtr hWnd,
            [Out] StringBuilder lpString, int nMaxCount);
[DllImport("user32.dll", SetLastError=true, CharSet=
                                        CharSet.Auto)]
public static extern int GetWindowTextLength(IntPtr
                                              hWnd);
```

Il secondo ci servirà per dimensionare correttamente la capacità dello *StringBuilder* da passare al metodo *GetWindowText*. Il primo è invece il metodo per ricavare il titolo di una finestra, dato il suo handle. Mediante una semplice interfaccia come quella mostrata in **Figura 2**, possiamo impostare in una *Textbox* direttamente l'handle della finestra, ricavabile mediante qualche altra API, o mediante qualche tool come *Spy++*.

GetWindowText handle 66592 this
Form1

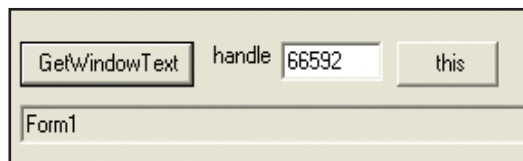
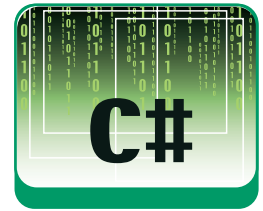
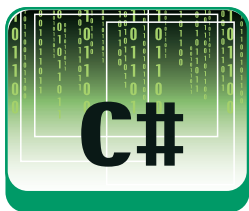



Fig. 2: **Ricavare il testo di una finestra dall'handle**

Il gestore del pulsante *GetWindowText* sarà così fatto:

```
private void btGetWindowText_Click(object sender,
                                  System.EventArgs e)
{
  try
  {
    IntPtr hWnd;
    if(txtHandle.Text.StartsWith("0x"))
      hWnd=new IntPtr(
              Convert.ToInt32(txtHandle.Text,16));
    else
      hWnd=new IntPtr(int.Parse(txtHandle.Text));
      int length = Win32.GetWindowTextLength(
                                          hWnd);
      StringBuilder sb = new StringBuilder(length + 1);
      int result=Win32.GetWindowText(hWnd, sb,
                                     sb.Capacity);
    if(result>0)
      txtWindowText.Text=sb.ToString();
    else
    {
      int error=Marshal.GetLastWin32Error();
      MessageBox.Show("errore: "+error);
    }
  }
  catch(Exception ex)
  {
    MessageBox.Show(ex.Message);
```

I TUOI APPUNTI

```
    }
}
```

Se provate ad inserire un handle non valido, ad esempio 0 oppure un valore troppo grande per un int, la funzione restituirà un errore, ricavabile mediante il metodo *Marshal.GetLastWin32Error.*
Un codice numerico però non è sufficiente per far capire all'utente quale errore si sia verificato, sarebbe meglio ottenere una descrizione testuale. Le API di windows forniscono a tale scopo la funzione *FormatMessage.* Tramite essa vediamo come è possibile implementare un semplice tool per ricavare da un errore numerico il relativo messaggio testuale.

```
//flags per FormatMessage
const int FORMAT_MESSAGE_ALLOCATE_BUFFER =
                                    0x00000100;
const int FORMAT_MESSAGE_IGNORE_INSERTS  =
                                    0x00000200;
const int FORMAT_MESSAGE_FROM_SYSTEM     =
                                    0x00001000;
[DllImport("kernel32.dll")]
static extern int FormatMessage(int dwFlags, IntPtr
          lpSource, int dwMessageId, int dwLanguageId,
          ref string lpBuffer, int nSize, IntPtr Arguments);
```

Dato l'errore *nError,* a questo punto basterà utilizzare il metodo in questa maniera:

```
string str="";
int dwChars= FormatMessage(
                  FORMAT_MESSAGE_ALLOCATE_BUFFER |
                    FORMAT_MESSAGE_FROM_ SYSTEM |
                   FORMAT_MESSAGE_IGNORE_INSERTS,
             IntPtr.Zero, nError, 0,  ref str, 0, IntPtr.Zero);
if(dwChars>0)
txtErrorMessage.Text=str;
else
      txtErrorMessage.Text="error not found";
```

WIN32	Linguaggio C	.NET (CTS)
HANDLE	void*	System.IntPtr
BYTE	Unsigned char	System.Byte
SHORT	Short	System.Int16
WORD	Unsigned short	System.UInt16
INT	Int	System.Int32
UINT	Unsigned int	System.UInt32
LONG	Long	System. Int32
BOOL	Long	System. Int32
DWORD	Unsigned long	System.UInt32
ULONG	Unsigned long	System.UInt32
CHAR	char	System.Char
LPSTR	char*	System.String oppure StringBuilder
LPCSTR	const char*	System.String oppure StringBuilder
LPWSTR	wchar_t*	System.String oppure StringBuilder
LPCWSTR	const wchar_t*	System.String oppure StringBuilder
FLOAT	float	System.Single
DOUBLE	double	System.Double

Tabella 1: PInvoke Data Types

La parte interessante di questo codice è il modo in cui si ottiene la stringa, che viene passata per riferimento, e quindi viene impostata all'interno della funzione nativa. In realtà, come detto nel caso di *GetLastError,* anche la funzione *FormatMessage* potrebbe comportarsi in maniera errata, ed esiste una maniera totalmente managed per ottenere lo stesso risultato, dunque il metodo precedente è servito essenzialmente come esercizio di pinvoke!
Conoscendo il codice di errore, ottenere il relativo messaggio testuale è cosa molto semplice, basta utilizzare la classe *Win32Exception,* passando l'errore al suo costruttore e leggendo poi la proprietà *Message:*

```
Win32Exception ex=new Win32Exception(error);
txtErrorMessage.Text=ex.Message;
```

In **Figura 3** potete vedere l'applicazione .NET *Error Lookup* in azione.

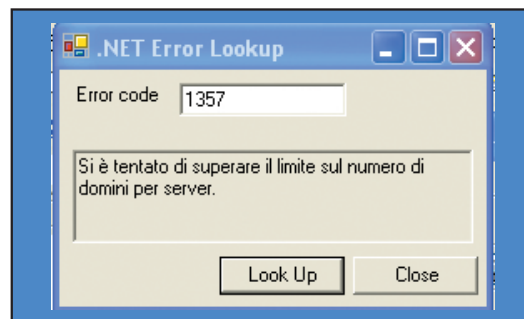



Fig. 3: L'applicazione .NET Error Lookup

MARSHALLING DI STRUTTURE E CLASSI

All'interno di una dll Win32, possono essere contenute anche strutture e classi. Anche queste necessitano naturalmente di essere tradotte nei corrispondenti tipi .NET. Infatti, il framework .NET non contiene una definizione managed di ogni tipo Win32, tantomeno potrebbe comprendere il formato di un tipo user-defined. Per utilizzare quindi un tipo struct nativo, in .NET, è necessario creare in un linguaggio managed una *struct* oppure una classe. Quando si va a definire una struttura che sia la versione managed di una struttura nativa, è necessario utilizzare l'attributo *StructLayout*, in maniera da informare il servizio *PInvoke* come trattare e come rappresentare i singoli membri della struttura. I possibili valori che può assumere l'attributo StructLayout sono quelli dell0enumerazione LayoutKind, e cioè uno fra *Auto, Explicit,* e *Sequential.* In generale si utilizza *Sequential,* per fare in modo che venga preservato l'ordine dei campi originali. Data ad esempio la struttura *RECT* rappresentante un rettangolo, definita come segue in *windef.h:*

```
typedef struct tagRECT
{  LONG    left;
   LONG    top;
   LONG    right;
   LONG    bottom;
} RECT
```

La corrispondente versione managed C#, potrebbe essere scritta così:

```
[StructLayout(LayoutKind.Sequential)]
public struct RECT
{  public int left;
   public int top;
   public int right;
   public int bottom;
}
```

Naturalmente questo è un caso molto semplice, è bastato trasformare il tipo *LONG* in *int*, e mantenere l'ordine sequenziale, ma facendo riferimento alla **Tabella 1**, basterà utilizzare i tipi giusti al posto giusto. Utilizzando invece il valore *Auto* come *StructLayout* si permetterà al CLR di riordinare i campi a suo piacimento, mentre con *Explicit* sarà possibile definire, per mezzo dell'attributo *FieldOffset* la posizione di ogni campo all'interno della struttura, calcolandola per mezzo della dimensione in byte del campo stesso. Ad esempio la precedente struttura *RECT* potrebbe essere scritta:

```
[StructLayout(LayoutKind.Explicit)]
public struct RECT
{ [FieldOffset(0)]
   public int left;
   [FieldOffset(4)]
   public int top;
   [FieldOffset(8)]
   public int right;
   [FieldOffset(12)]
   public int bottom;
}
```

Ogni campo int ha infatti una dimensione di 4 byte. Dunque il secondo byte sarà posizionato a 4 di distanza dal primo, il terzo a 8, ed il quarto a 12.

CATTURARE LO SCHERMO

Dopo aver parlato dei vari aspetti di PInvoke, vediamo come utilizzarli all-in-one per creare un'applicazione che necessiti di ricorrere alle API Win32, ad esempio per catturare il desktop, una finestra, o un'area dello schermo, e salvare l'immagine ottenuta su file. Le funzioni necessarie in questo caso saranno contenute nelle librerie *GDI32.dll* e *User32 .dll*. In questo caso sono state quindi create le due classi *GDI32* e *User32*, che in generale hanno questo aspetto:

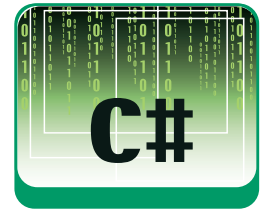
```
class User32
{ [DllImport("user32.dll")]
   public static extern IntPtr GetDesktopWindow();
   [DllImport("user32.dll")]
   public static extern IntPtr WindowFromPoint(Point pt);
   //... altre funzioni, struct, costanti
}
```

Con i metodi delle due classi sarà un gioco da ragazzi catturare lo schermo, singole finestre, o rettangoli di qualunque dimensione e posizione. Ricordando che anche il desktop è una finestra come tutte le altre, il primo passo da fare è quello di ottenere l'handle della finestra da catturare. Per catturare l'intero desktop o una sua parte rettangolare niente di più semplice, basta utilizzare la funzione *GetDesktopWindow:*

```
[DllImport("user32.dll")]
public static extern IntPtr GetDesktopWindow();
```

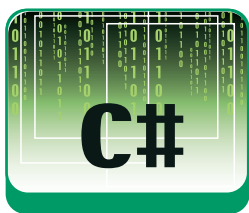
Dato l'handle, per ottenere un'oggetto *Image* dobbiamo far entrare in gioco le funzioni di *GDI32.dll.* Il seguente metodo realizza lo scopo:

```
public static Image CaptureRect(Rectangle windowRect)
{  IntPtr handle=User32.GetDesktopWindow();
    // ottiene il DC dall'handle
    IntPtr hdcSrc = User32.GetWindowDC(handle);
    int width = windowRect.Right - windowRect.Left;
    int height = windowRect.Bottom - windowRect.Top;
    // crea il DC su cui copiare l'immagine
    IntPtr hdcDest = GDI32.CreateCompatibleDC(
                                             hdcSrc);
    IntPtr hBitmap = GDI32.CreateCompatibleBitmap(
                                 hdcSrc,width,height);
    IntPtr hOld = GDI32.SelectObject(hdcDest,hBitmap);
    GDI32.BitBlt(hdcDest,0,0,width, height, hdcSrc,
            windowRect.X,windowRect.Y,GDI32.SRCCOPY);
    GDI32.SelectObject(hdcDest,hOld);
    // rilascia risorse
   GDI32.DeleteDC(hdcDest);
   User32.ReleaseDC(handle,hdcSrc);
   // costruisce un oggetto Image dall'handle
                                          della Bitmap
   Image img = Image.FromHbitmap(hBitmap);
   GDI32.DeleteObject(hBitmap);
   return img;
}
```

Riassumendo, i passi utilizzati sono quelli di ottenere il *Device Context* della finestra, calcolarne le dimensioni, creare una nuova bitmap,copiarci sopra l'immagine catturata tramite la funzione *BitBlt*, ed infine rilasciare le risorse. Catturare ora l'intero

NOTA

QUALI FUNZIONI CONTIENE UNA DLL

Se non si ha a disposizione la documentazione del platfom SDK di Windows, è possibile utilizzare l'utility dumpbin, da riga di comando, per ricavare le funzioni esportate da una qualsiasi dll. Ad esempio per ottenere le funzioni esportate dalla dll User32.dll, basta lanciare il comando:

```
dumpbin –exports
                 user32.dll
```

desktop o una finestra è assolutamente equivalente, basta invocare il metodo precedente, utilizzando gli adeguati rettangoli, per il desktop basta utilizzare direttamente l'handle ottenuto con *GetDesktopWindow,* mentre per una finestra bisognerà ricavare l'oggetto *RECT* che definisce la posizione e le dimensioni:

```
public static Image CaptureScreen()
{ return CaptureWindow( User32.GetDesktopWindow() );}
public static Image CaptureWindow(IntPtr handle)
{ User32.RECT windowRect = new User32.RECT();
  User32.GetWindowRect(handle,ref windowRect);
  Rectangle rect=new Rectangle(
                    windowRect.left,windowRect.top,
  windowRect.right - windowRect.left,
  windowRect.bottom - windowRect.top);
  return CaptureRect(rect);
}
```

Dato che ora abbiamo a disposizione tutti gli strumenti necessari, non resta che creare un'infrastruttura che permetta di utilizzarli nella maniera più user-friendly. Per catturare un'area rettangolare si può pensare di utilizzare l'area client dell'applicazione stessa come mirino, facendola diventare trasparente quando inviamo il comando di *Start Capture* e riportandola al suo stato originale una volta completato lo "scatto" dell'istantanea. La **Figura 4** mostra uno screenshot della nostra applicazione in modalità di cattura.

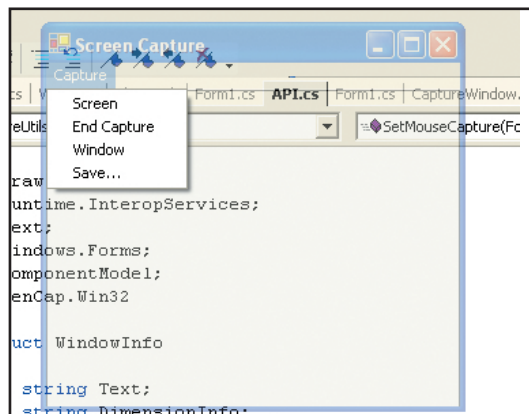



***Fig. 4:* Catturare un rettangolo del desktop**

La parte meno semplice, se non ci fosse pInvoke, sarebbe quella di selezionare la finestra che si vuol catturare ed ottenerne l'handle. Un modo sarebbe quello di ottenere l'elenco di tutte le finestre attive con relativi handle, ma non sarebbe il massimo dell'usabilità. Citando ancora Spy++, faremo in modo che la selezione della finestra avvenga tramite mouse, abilitando la modalità di cattura al click su un pulsante, e selezionando la finestra da catturare trascinando il puntatore su di essa. Infine l'handle sarà ricavato al rilascio del pulsante. La **Figura 5** mostra il diagramma degli stati dell'intera operazione. In questo caso, fondamentali sono le funzioni *SetMouseCapture* e *ReleaseCapture,* che permettono, la prima, di redirigere gli eventi del mouse in maniera da farli gestire alla nostra Form, e la seconda di disattivare la redirezione. Quindi al click del mouse attiveremo la modalità capture così:

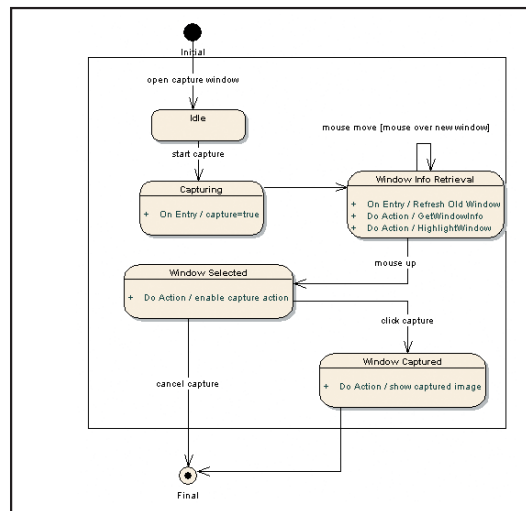



***Fig. 5:* Il diagramma degli stati per la cattura di una finestra**

```
private void button1_MouseDown(object sender,
              System.Windows.Forms.MouseEventArgs e)
{  CaptureUtils.SetMouseCapture(this);
   capture=true;
   this.Cursor=Cursors.Cross;
}
```

Durante il movimento del mouse, dovremo evidenziare la finestra attiva, quella cioè che si trova sotto il puntatore, e per far ciò dovremo innanzitutto ottenere tramite la funzione *WindowFromPoint* il suo handle, e da questo informazioni come il testo, e le dimensioni, in questa maniera:

```
IntPtr handle=User32.WindowFromPoint(
                  PointToScreen(new Point(e.X,e.Y)));
CaptureUtils.HighlightWindow(handle);
GetWindowInfo(handle);
```

Il metodo *HighlightWindow* non fa altro che disegnare un rettangolo di colore giallo come bordo del-

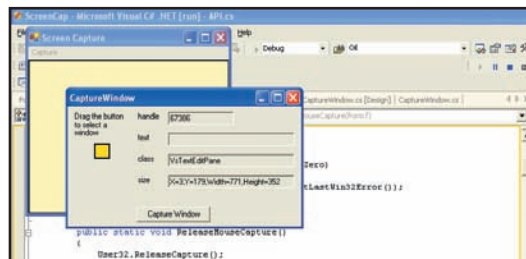



***Fig. 6:* Catturare una finestra con l'uso del mouse**

BIBLIOGRAFIA

• PROGRAMMING WINDOWS WITH C# *Petzold* (Microsoft Press)

• COM AND .NET PROGRAMMING *A. Troelsen* (Apress)

la finestra attiva, mentre il metodo *GetWindowInfo* ricava a partire dall'handle, il testo, il classname ed il rettangolo della finestra. La **Figura 6** mostra l'applicazione, completata ed in modalità di cattura. Come al solito rimandiamo al codice allegato per una miglior comprensione e studio.

FUNZIONI DI CALLBACK

Una funzione di *callback,* è una funzione implementata nello stesso codice di una funzione chiamante, e che sarà invocata però all'interno di una data funzione chiamata, in maniera da poter interagire con il chiamante. Per far ciò viene passato come argomento alla funzione un puntatore alla funzione di callback. Per utilizzare tale tipo di funzioni in .NET è necessario ricorrere al concetto di delegate, cioè il corrispondente nel mondo managed dei puntatori a funzione unmanaged. Una funzione che necessita ad esempio di un puntatore a funzione da invocare come *callback* è la *EnumWindows,* che permette di ottenere tutte le finestre attive sullo schermo, e che ha questa firma in C:

```
BOOL EnumWindows(WNDENUMPROC lpEnumFunc,
                                     LPARAM lParam)
```

Il primo argomento è un puntatore ad una funzione che deve avere questa sintassi:

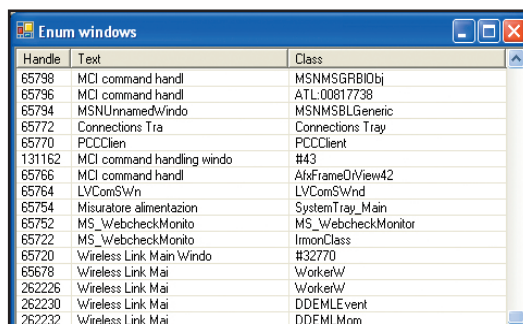
```
BOOL CALLBACK EnumWindowsProc(
                         HWND hwnd, LPARAM lParam));
```

Il primo argomento sarà l'handle di un finestra, mentre il secondo dipende dall'applicazione, e per il nostro esempio possiamo anche trascurarlo. In C# dovremo quindi innanzitutto definire un delegate corrispondente a tale funzione di callback:

```
public delegate bool EnumWindowsCallBack( IntPtr
                              handle, IntPtr param );
```

Mentre per importare la funzione *EnumWindows* tramite *DllImport* scriveremo:

```
[DllImport("user32.dll")]
public static extern bool EnumWindows(
              EnumWindowsCallBack cb, IntPtr param );
```

per utilizzare il metodo *EnumWindows* non ci resta che creare un metodo che implementi la firma del delegate, ad esempio se vogliamo stampare l'handle, il titolo, e il nome della classe di ogni finestra aperta sullo schermo, scriveremo il metodo seguente:

```
private static bool EnumWindowsProc( IntPtr handle,
                                        IntPtr param )
{ Console.WriteLine("handle: {0}", handle );
  int len=LibWrap.GetWindowTextLength(handle);
  StringBuilder sb=new StringBuilder(len);
  LibWrap.GetWindowText(handle,sb,len);
  Console.WriteLine("text: {0}",sb);
  sb=new StringBuilder(1024);
  LibWrap.GetClassName(handle,sb,1024);
  Console.WriteLine("class: {0}",sb);
  Console.WriteLine("--------------");
  return true;
}
```

L'ultimo passo è istanziare il delegate e passarlo al metodo *EnumWindows,* così:

```
EnumWindowsCallBack callback = new
         EnumWindowsCallBack( EnumWindowsProc );
LibWrap.EnumWindows(callback, IntPtr.Zero );
```

Integrando il codice in un'applicazione *Windows Forms* (vedi codice allegato), ad esempio con una *ListView* per inserire i dati delle finestre, otterrete un risultato analogo a quello in **Figura 7**.

Enum windows

Handle	Text	Class
65798	MCI command handl	MSNMSGRBIObj
65796	MCI command handl	ATL:00817738
65794	MSNUnnamedWindo	MSNMSBLGeneric
65772	Connections Tra	Connections Tray
65770	PCCClien	PCCClient
131162	MCI command handling windo	#43
65766	MCI command handl	AfxFrameOrView42
65764	LVComSWn	LVComSWnd
65754	Misuratore alimentazion	SystemTray_Main
65752	MS_WebcheckMonito	MS_WebcheckMonitor
65722	MS_WebcheckMonito	IrmonClass
65720	Wireless Link Main Windo	#32770
65678	Wireless Link Mai	WorkerW
262226	Wireless Link Mai	WorkerW
262230	Wireless Link Mai	DDEMLEvent
262232	Wireless Link Mai	DDEMLMom

Fig. 7: **Le finestre aperte sul mio pc sono forse troppe?**

CONCLUSIONI

In questo articolo abbiamo mostrato come interagire con le funzioni unmanaged e native del sistema operativo, all'interno di un progetto *managed.* La tecnologia PInvoke consente dunque di continuare ad utilizzare il codice *legacy* che ancora non è stato migrato su .NET, in attesa naturalmente della migrazione totale!

Antonio Pelleriti

C#

CONTATTA L'AUTORE

Potete rivolgere domande di chiarimenti o ulteriori richieste all'autore all'indirizzo antonio.pelleriti@ioprogrammo.it, o ancora sul forum di ioProgrammo forum.ioprogrammo.net o sul sito www.dotnetarchitects.it

VB.NET E PINVOKE

In Vb.NET, come sarà familiare ai programmatori vb6, è possibile importare la funzione anche mediante l'istruzione declare, ma per fornire alcune impostazioni è ancora necessario l'utilizzo di DllImport. Ad esempio sono possibili entrambe le seguenti sintassi:

```
Declare Auto Function Beep Lib
    "kernel32.dll" (ByVal freq As Integer,
              ByVal duration As Integer)
                              As Integer
<DllImport("kernel32.dll")>
Public Shared Function Beep( ByVal
      freq As Integer, ByVal duration
              As Integer) As Integer
End Function
```

Gestisci i tuoi dati con Java e iBatis

L'utilizzo del framework iBatis può incrementare notevolmente la facilità e la velocità di sviluppo di applicazioni che interagiscono con un DB, vediamo come

J2SE

REQUISITI

Conoscenze richieste
Java, SQL, XML

Software
JDK 1.4, iBatis 2.0 e Eclipse 3.0

Impegno

Tempo di realizzazione

La persistenza dei dati trattati da un'applicazione è sempre stata un punto cruciale durante il processo di sviluppo del software. Il programmatore deve sempre districarsi tra le API messe a disposizione dal linguaggio di programmazione che utilizza ed il modo in cui memorizza i dati. Nella maggior parte dei casi però si ricorre all'utilizzo di un RDBMS, ciò significa che la persistenza dei dati viene effettuata utilizzando un approccio relazionale; esso prevede la memorizzazione dei dati su tabelle. Il linguaggio Java, essendo un linguaggio Object Oriented, focalizza la sua attenzione sul concetto di oggetto. Java espone le API JDBC per adattare la propria natura ad oggetti con il concetto di tabelle degli RDBMS; queste API consentono di effettuare qualsiasi operazione su un qualunque RDBMS per il quale esiste un'implementazione Java del driver. Nel corso di questo articolo studieremo i caratteri generali di un framework open source che facilita l'utilizzo di basi di dati relazionali con il linguaggio Java. Il framework di cui parleremo è iBatis ed essendo un software open source molto utilizzato è facile reperire in rete tutorial, articoli ed esempi di utilizzo. Per meglio comprendere le sue peculiarità andremo ad implementare la parte di gestione dei dati *(Data Layer)* di una semplice applicazione per la gestione dei dipendenti di un'azienda. Nella parte finale di questo articolo vedremo come iBatis si sposi con il pattern DAO, molto utilizzato dai moderni progettisti di software.

obj1.xml
SqlMapConfig.xml
obj2.xml
Parametri di Input
- JavaBeans
- Map
- Tipi primitivi

SQL Maps
PreparedStatement
PreparedStatement
PreparedStatement

Valori di Output
- JavaBeans
- Map
- List
- Tipi primitivi

JDBC API
DB

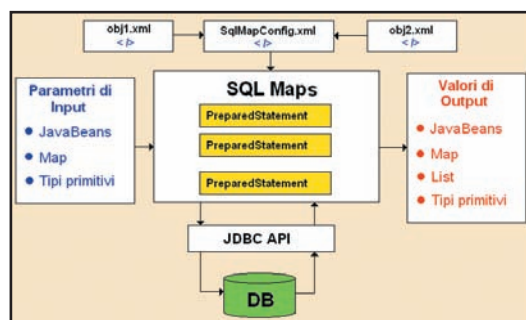
Fig. 1: **Lo schema funzionale del framework SQL Maps**

COSA SERVE PER INIZIARE

Dopo aver scaricato iBatis 2.0 all'URL ibatis .apache.org, scompattarlo in locale sul proprio hard disk; includere i file *commons-logging-1-0-3.jar, ibatis-common-2.jar, ibatis-sqlmap-2.jar* nel classpath della propria applicazione Java. Scaricare HSQLDB all'URL www.dadasd .sdsad, scompattarlo in locale sul proprio hard disk ed includere il file *hsqldb.jar* nel classpath della propria applicazione Java.

PERCHÈ IBATIS?

Da sempre il mondo degli sviluppatori Java è uno di quelli che trae più profitto dai progetti open source; anche per questo motivo diminuiscono sempre di più i programmatori che utilizzano direttamente le API JDBC. Possiamo affermare con una certa sicurezza che tra tutti i framework inerenti l'accesso a basi di dati relazionali, quello che ha sbaragliato la concorrenza è senza dubbio Hibernate. Questo tool insieme agli EJB entity e alla tecnologia JDO è il più noto tra i cosiddetti ORM tools *(Object to Relational Mapping tools)*, ciò non significa che in tutti i contesti ne vengano decantate le lodi. In molte applicazioni lo sviluppo del Data Layer è spesso affidato a team di persone composte anche da DBA *(database administrator)*, i quali sono molto diffidenti da tool che generano automaticamente le query SQL.
Inoltre lo sviluppatore spesso manifesta questa perplessità: "*perchè devo imparare una variante dell'SQL come HSQL o JDOQL per interrogare un database relazionale?*" iBatis risolve questi dubbi consentendo l'utilizzo dell'SQL per l'interrogazione di un RDBMS, inoltre è molto più facile da impiegare ed apprendere rispetto ai più diffusi ORM tools. Ogni applicazione che utilizza JDBC deve occuparsi di:

1) Gestire le connessioni e le transazioni
2) Estrarre da oggetti Java i parametri per le query

SQL
3) Creare le query SQL
4) Convertire i risultati ottenuti da ResultSet nell'oggetto Java appropriato

iBatis definisce un framework chiamato SQLMaps che riduce sensibilmente il codice Java che viene utilizzato per implementare i precedenti punti. Si occupa in modo molto minuzioso sia del mapping tra i JavaBean dell'applicazione ed i parametri dei *PreparedStatement,* sia del processo inverso, ovvero del mapping tra i ResultSet e gli oggetti Java. L'idea di fondo che sta dietro l'impiego di SQLMaps è quella di creare uno o più file XML che contengono le query SQL che si intendono utilizzare in un'applicazione. Per ogni query bisogna specificare il mapping tra i parametri in input alla query stessa e l'oggetto Java che li contiene, inoltre bisogna specificare il mapping tra il risultato della query e gli oggetti Java da popolare. Infatti quando si esegue una query, bisogna creare un oggetto che contiene i parametri da inserire nel corrispondente *PreparedStatement* e passarlo insieme al nome della query all'oggetto. Eseguita la query, SQLMaps creerà una o più istanze della classe specificata nel file XML, riempendole con i dati recuperati dal ResultSet. È inutile sottolineare che questo approccio necessita della conoscenza di fondo dell'SQL e delle regole fondamentali inerenti la creazione di file XML.

UN'APPLICAZIONE D'ESEMPIO

Andiamo adesso a definire le caratteristiche principali dell'applicazione di esempio che utilizzeremo per imparare sul campo l'impiego di questo potente tool. Supponiamo di voler gestire con una semplice applicazione il personale di un'azienda, a questo scopo memorizzeremo i dati su un DB relazionale e precisamente su due tabelle; seguendo un'architettura a livelli, suddividiamo l'applicazione in più layer (presentation layer, business logic e data layer per esempio) e ci concentreremo sul data layer. Così facendo non faremo dipendere la gestione dei dati dalla loro presentazione, potremmo utilizzare Swing o tecnologie web come le JSP senza cambiare nulla nel nostro layer. Iniziamo a definire i dati che si vogliono gestire senza tener conto della loro rappresentazione per la persistenza.
Per semplicità un dipendente sarà caratterizzato dai seguenti dati:

- **iddip** (identificativo univoco del dipendente)
- **nome**
- **cognome**
- **mansione** (*Dirigente, Quadro, Impiegeto, Operaio*)
- **sistemazione** (il posto in cui lavora).

Mentre una sistemazione conterrà i seguenti dati:

- **idsis** (identificativo univoco della sistemazione)
- **label** (etichetta affissa sul locale)
- **piano**
- **tipo** (ufficio, magazzino, laboratorio)

Nell'implementare gli oggetti che conterranno questi campi, terremo presente il pattern *Value Object,* il quale mira ad alleggerire il più possibile i dati scambiati tra i vari layer di un'applicazione. Questo accorgimento in ambienti distribuiti aiuta a diminuire il traffico di rete e quindi ad aumentare l'efficienza del sistema.

```
public class SistemazioneVO
implements Serializable{
   public static final int UFFICIO=0;
   public static final int LABORATORIO=1;
   public static final int MAGAZZINO=2;
   protected int idSis=-1;
   protected String label=null;
   protected int piano=0;
   protected int tipo=UFFICIO;
   public int getIdSis() {return idSis;}
   public void setIdSis(int idUff) {this.idSis = idUff;}
   public String getLabel() {return label;}
   public void setLabel(String label) {
      this.label = label;
   }
   public int getPiano() {return piano;}
   public void setPiano(int piano) {
     this.piano = piano;
   }
   public int getTipo() { return tipo;}
   public void setTipo(int tipo) {this.tipo = tipo;}
}
```

Come possiamo notare la classe rappresenta un normale JavaBean munito esclusivamente di metodi *set* e *get.* Un approccio del tutto uguale viene utilizzato per l'implementazione della classe *DipendenteVO:*

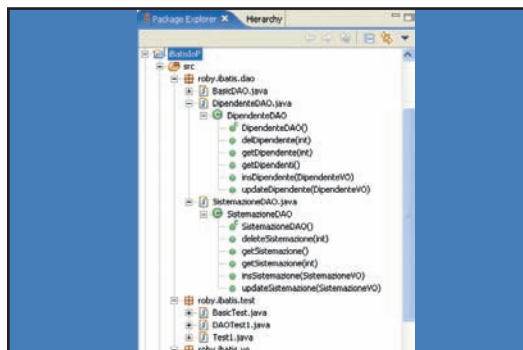



Fig. 2: La struttura del progetto vista da Eclipse

I TUOI APPUNTI

NOTA

IBATIS SOLO IN JAVA?

Il framework iBatis è un prodotto open source rilasciato sotto licenza Apache 2.0, ha riscosso tanto successo nel mondo degli sviluppatori che esiste una versione anche per .NET; essa facilita l'utilizzo di ADO.NET.

```
public class DipendenteVO implements
Serializable {
    public static final int DIRIGENTE=3;
    public static final int QUADRO=2;
    public static final int IMPIEGATO=1;
    public static final int OPERAIO=0;
    protected int idDip=-1;
    protected String nome=null;
    protected String cognome=null;
    protected int mansione=OPERAIO;
    protected SistemazioneVO sistemazione=null;
    public SistemazioneVO getSistemazione() {
        return sistemazione; }
    public void setSistemazione(
                        SistemazioneVO sistemazione) {
        this.sistemazione = sistemazione;
    }
    ....
    // altri metodi set e get delle proprietà
    ....
}
```

IL CUORE DI IBATIS È SCRITTO IN XML

Ogni applicazione che utilizza iBatis deve leggere un file XML che rappresenta la base per l'integrazione e l'interazione dell'oggetto SQLMapClient con uno degli RDBMS supportati. Nel nostro esempio utilizzeremo HSQLDB, il passaggio ad un altro qualsiasi RDBMS è immediato ed intuitivo. Poichè il file XML serve a fornire la modalità di funzionamento di un SQLMapClient lo chiameremo *SqlMapConfig.xml*. Nella sua definizione sono previste delle sezioni opzionali e delle sezioni obbligatorie, per motivi di spazio e vista la natura didattica di questo articolo utilizzeremo una configurazione minima ma funzionante. Il Data Layer implementato in questo articolo utilizzerà un file XML cosi strutturato:

```
<sqlMapConfig>
<transactionManager type="JDBC">
<dataSource type="SIMPLE">
<property name="JDBC.Driver"
value="org.hsqldb.jdbcDriver"/>
<property name= "JDBC.ConnectionURL"
value="jdbc:hsqldb:file:/java
            /iBatisTestIoP/data/azienda"/>
<property name="JDBC.Username"
                            value="sa"/>
<property name="JDBC.Password"
                            value=""/>
</dataSource>
</transactionManager>
<sqlMap resource="Dipendente.xml"/>
<sqlMap resource="Sistemazione.xml"/>
</sqlMapConfig>
```

Il tag transactionManager serve a definire il tipo di transazione utilizzata per l'accesso al DB, JDBC è il più semplice in quanto utilizza direttamente le primitive offerte dal package java.sql. Altri valori utilizzabili sono JTA ed EXTERNAL e servono a definire rispettivamente l'utilizzo della tecnologia JTA e una possibile implementazione customizzata delle transazioni.
Il tag dataSource e tutti i tag in esso annidati servono per definire la gestione delle connessioni al DB e quindi il tipo di pooling da utilizzare. Il valore SIMPLE significa che verrà utilizzata un'implementazione proprietaria di iBatis. Altri possibili valori sono DBCP, per utilizzare l'omonima libreria open source rilasciata da Apache, e JNDI che rimanda la gestione delle connessioni ad un container esterno che si basa appunto su JNDI. Il valore *jdbc:hsqldb:file:/java/iBatisTestIoP/data/azienda* rappresenta il path assoluto dei file che contengono il database creato con HSQLDB.
Se si istalla l'applicazione su un path diverso da */java /iBatisTestIoP/* bisogna sostituirlo a questo valore presente nel file XML.
Alla fine del file *SqlMapConfig.xml* possono essere definiti diversi tag sqlMap, questi servono ad elencare i file XML in cui sono contenute le query SQL e gli opportuni mapping tra tabelle ed oggetti Java.

QUERY SQL IN XML?

Il più grande vantaggio nell'utilizzare iBAtis, e particolarmente il framework SQLMaps, è quello di poter gestire le query SQL all'interno di file XML. Questo approccio è molto proficuo in quanto permette di gestire le query SQL separatamente dal codice Java, consentendo modifiche sostanziali senza moficare i sorgenti e ricompilarli. La funzionalità di questi file XML non finisce qui, infatti essi si occupano anche del mapping tra oggetti Java e parametri SQL nonchè del mapping tra risultati delle query ed oggetti Java. Per meglio comprendere quanto detto, prendiamo in esame il file *Sistemazione.xml*, esso contiene le query per la gestione degli oggetti *SistemazioneVO*.

```
<delete id="deleteSistemazione" parameterClass=
                            "java.lang.Integer">
delete from Sistemazione where idsis=#value#
</delete>
```

Il tag suggerisce che la query contenuta in esso effettua una cancellazione sul DB; in particolare il valore dell'attributo id determina la stringa utilizzata nel codice Java per identificare l'enunciato SQL in questione. L'attributo *parameterClass* invece specifica la classe del parametro passato alla query, il suo valore viene utilizzato dal framework per sostituire la stringa *#value#* nella query stessa. Dopo questo primo assaggio, possiamo facilmente immaginare che il framework preveda tanti tipi di tag per quanti sono gli enunciati SQL. Nello stesso file possiamo notare anche il seguente tag:

```
<update id="updateSistemazione" parameterClass=
                    "roby.ibatis.vo.SistemazioneVO">
update Sistemazione
set label=#label#,
piano=#piano#,
tipo=#tipo#
where idsis=#idSis#
</update>
```

In questo caso il parametro passato alla query non è più istanza di una classe appartenente al package java.lang come succedeva nel caso precedente; per questo motivo possono essere utilizzati nella query tutti i suoi campi muniti di metodi *getXXX*. Osserviamo il tag destinato al recupero dei dati dal DB per mezzo dell'enunciato *select:*

```
<select id="getSistemazione" parameterClass=
                            "java.lang.Integer"
resultClass="roby.ibatis.vo.SistemazioneVO">
select * from Sistemazione where idsis=#value#
</select>
```

Rispetto agli esempi precedenti viene introdotto un

nuovo attributo; *resultClass* serve a mappare il risultato della query su un oggetto Java istanza della classe *SistemazioneVO*. Il mapping è esplicito in quanto i nomi dei campi della tabella *Sistemazione* corrispondono alle proprietà munite di metodi *setXXX* della classe *SistemazioneVO*. Questo meccanismo evita la gestione diretta di un *ResultSet* come prevedono le API JDBC.

Fig. 3: ibatis.apache.org il punto di partenza per approfondire la conoscenza di iBatis

MAPPING AVANZATO

Adesso aumentiamo un po' la complessità analizzando il tag che definisce l'operazione di inserimento sul DB.

```
<insert id="insSistemazione" parameterClass=
                        "roby.ibatis.vo.SistemazioneVO">
insert into Sistemazione(label,piano,tipo)
values(#label#,#piano#,#tipo#)
<selectKey resultClass="int" keyProperty="idSis">
select max(idsis) from sistemazione
</selectKey>
</insert>
```

La prima parte non presenta nessuna novità in quanto conosciamo già il significato dell'attributo id e quello dell'attributo *parameterClass*. Anche il mapping dei parametri della query è facile da intuire, la novità invece risiede nell'utilizzo del tag *selectKey*. Questo tag può essere impiegato solo all'interno del tag *insert;* esso viene utilizzato per valorizzare gli attributi degli oggetti che mappano i campi autoincrement di una tabella. In questo caso specifico serve a valorizzare l'attributo *idSis* dell'oggetto istanza di *SistemazioneVO* passato alla query.
Fino a questo punto abbiamo esaminato query SQL in cui vengono utilizzati oggetti semplici, cioè oggetti i cui campi, forniti di metodi *set* e *get*, sono rappresentati da tipi predefiniti, istanze di classi appartenenti al package *java.lang* o da stringhe. Aumentiamo ancora un po' la complessità e prendiamo in esame il file *Dipendente.xml* in cui vengono gestite le query inerenti il corrispondente oggetto *DipendenteVO;* questo oggetto composto contiene l'attributo sistemazione istanza della classe *SistemazioneVO*. Per ovviare a questa situazione leggermente più complessa SQLMaps prevede la dichiarazione tag come *parameterMap* e *resultMap:*

```
<parameterMap id="dipendentePM" class=
                        "roby.ibatis.vo.DipendenteVO">
  <parameter property="nome" />
  <parameter property="cognome" />
  <parameter property="mansione" />
  <parameter property="sistemazione.idSis" />
</parameterMap>
<resultMap id="dipendenteRM" class=
                        "roby.ibatis.vo.DipendenteVO">
   <result property="idDip" column="iddip"/>
   <result property="nome" column="nome"/>
   <result property="cognome" column="cognome"/>
   <result property="mansione" column="mansione"/>
   <result property="sistemazione.idSis" column=
                                        "idsis"/>
   <result property="sistemazione.label" column=
                                        "label"/>
   <result property="sistemazione.piano" column=
                                        "piano"/>
   <result property="sistemazione.tipo" column=
                                        "tipo"/>
</resultMap>
```

Un tag *parameterMap* serve a valorizzare i parametri di una query con oggetti contenuti in altri oggetti; i valori definiti dai tag *parameter* in esso innestati vengono mappati nella query nello stesso ordine in cui sono dichiarati. In questo caso l'attributo id definisce il valore utilizzato dall'attributo *parameterMap* in un tag *insert*.

```
<insert id="insDipendente" parameterMap=
                                "dipendentePM">
insert into dipendente(
       nome,cognome,mansione,xidsis) values(?,?,?,?)
<selectKey resultClass="int" keyProperty="idDip">
select max(iddip) from dipendente
</selectKey>
</insert>
```

In questa query i quattro "?" vengono sostituiti con i valori recuperati utilizzando il mapping definito nel tag *parameterMap;* in particolare l'ultimo sarà valorizzato con il campo *idSis* dell'oggetto sistemazione recuperato da un oggetto *DipendenteVO* (dietro le quinte viene utilizzato *dip.getSistemazione().getIdSis()*). Analizziamo invece il seguente codice XML per comprendere l'utilizzo del tag *resultMap:*

```
<select id="getDipendenti" parameterClass=
                              "java.lang.Integer"
resultMap="dipendenteRM">
```

NOTA

ESEGUIRE L'APPLICAZIONE

Per vedere funzionare l'applicazione è sufficiente, dopo aver modificato il file SqlMapConfig.xml come spiegato nel corso dell'articolo, eseguire il file DAOTest1.bat o Test1.bat sotto la directory iBatisTestIoP/bin.

```
select * from dipendente,
sistemazione where sistemazione.idsis=
                                    dipendente.xidsis
</select>
```

In questo caso, il framework SQLMaps, eseguita la query e recuperati i dati dal DB, istanzia n oggetti *DipendenteVO* e ne valorizza le proprietà utilizzando gli abbinamenti definiti nel tag *resultMap*. In particolare l'attributo XML *property* definisce la proprietà dell'oggetto Java da valorizzare con il campo (colonna della tabella) specificato dall'attributo *column*. Anche per l'attributo *property* viene utilizzata la notazione col punto, per esempio:

```
<result property="sistemazione.idSis" column=
                                          "idsis"/>
```

corrisponde al seguente codice Java:

```
DipendenteVO dip=new DipendenteVO();
dip.setSistemazione( new SistemazioneVO());
int id= rs.getInt("idsis");
dip.getSistemazione().setIdSis(id);
```

NOTA

IBATIS CARATTERISTICHE AVANZATE

In questo articolo vengono presentate le caratteristiche principali di iBatis. In realtà questo framework presenta altre caratteristiche più sofisticate come: gestione delle transazioni *(start, commit, rollback)*, caching dei dati e paginazione dei risultati.

UTILIZZARE SQLMAPS

A questo punto abbiamo tutte le conoscenze necessarie per effettuare operazioni sul DB sfruttando iBatis e precisamente SQLMaps. Per interagire con un RDBMS, utilizzando iBatis, è necessario, oltre ad aver definito i file XML finora analizzati, utilizzare un oggetto Java istanza della classe *SqlMapClient*.
A tal proposito osserviamo il contenuto del metodo *main* della classe *Test1*.

```
SqlMapClient sqlMap=null;
Reader reader = Resources. getResourceAsReader
                                ("SqlMapConfig.xml");
sqlMap = SqlMapClientBuilder.buildSqlMapClient(
                                          reader);
```

La sintassi è molto semplice in quanto bisogna ottenere un oggetto che implementa *java.io.Reader* dal metodo statico *getResourceAsReader* di *Resources*. Questo metodo accetta un parametro stringa che specifica un file XML contenuto nel classpath dell'applicazione, il file contiene la configurazione *(vedi apposito Riquadro)* del framework SQLMaps.
Una volta ottenuto il *Reader* basta passarlo al metodo statico *buildSqlMapClient* di *SqlMapClientBuilder*. L'oggetto *sqlMap* può essere utilizzato come un singleton in quanto al suo interno gestisce in modo trasparente le connessioni al DB.
Con in mano l'oggetto *sqlMap* possiamo effettuare una delle operazioni definite nei file *Dipendente.xml* e *Sistemazione.xml*.

```
SistemazioneVO sist1= null;
sist1= initSis( "I0-01", 0, SistemazioneVO.UFFICIO);
SistemazioneVO sist2= null;
sist2= initSis( "I0-02", 0, SistemazioneVO.UFFICIO);
sqlMap.insert("insSistemazione",sist1);
sqlMap.insert("insSistemazione",sist2);
```

Una volta inizializzati i due oggetti istanza di *SistemazioneVO*, è sufficiente invocare il metodo insert dell'oggetto *sqlMap* per effettuare l'omonimo enunciato SQL. Notiamo come il parametro stringa *"insSistemazione"*, passato al metodo *insert*, corrisponde al valore dell'attributo *id* presente nel tag

```
<insert id="insSistemazione" parameterClass=
                          "roby.ibatis.vo.SistemazioneVO">
...
</insert>
```

del file *Sistemazione.xml*. Il secondo parametro corrisponde invece ad un oggetto istanza della classe definita dall'attributo *parameterClass*.
Operazioni molto simili possono essere effettuate per quanto riguarda oggetti di tipo *DipendenteVO:*

```
DipendenteVO roby= initDip( "Roberto", "Sidoti",
                              DipendenteVO.IMPIEGATO);
roby.setSistemazione(sist1);
sqlMap.insert("insDipendente",roby);
```

La particolarità di questa operazione è che attraverso il meccanismo descritto in precedenza (vedi tag *parameterMap*) è possibile utilizzare nella query sia l'oggetto istanza di *DipendenteVO* che l'oggetto istanza di *SistemazioneVO* in esso contenuto.

```
DipendenteVO bill= ....
sqlMap.insert("insDipendente",bill);
roby.setMansione(DipendenteVO.DIRIGENTE);
sqlMap.update( "updateDipendente", roby);
sqlMap.delete("delDipendente",
       new Integer( bill.getIdDip() ) );
```

In quest'altra porzione di codice osserviamo l'impiego dei metodi update e delete; la sintassi è uguale a quella del metodo insert visto in precedenza.
A questo punto potremmo chiederci: come si effettuano query per il recupero dei dati con iBatis?
Il framework *SQLMaps* ha pensato proprio a tutto, infatti tra i vari metodi che mette a disposizione dei programmatori presenta *queryForObject* e *queryForList*. Il primo serve a recuperare un singolo oggetto dal DB, mentre il secondo, il cui impiego analizzeremo di seguito, serve a recuperare una lista di oggetti dalla base di dati.

```
List si= sqlMap.queryForList("getSistemazioni",null);
ArrayList sistemazioni=new ArrayList( si );
```

Come al solito il primo parametro serve ad identificare la query all'inteno dei file XML, il secondo parametro in questo caso è inutile quindi è possibile avvalersi di un valore nullo. Adesso che conosciamo molte delle caratteristiche di iBatis possiamo notare come il framework sia molto flessibile a cambiamenti apportati sui dati da gestire. Infatti ipotizziamo di dover aggiungere il campo eta alla tabella dipendnte, le modifiche da apportare sono minime in quanto basta modificare la classe *DipendenteVO* ed in minima parte il file *sistemazione.xml.*

Fig. 4: **hsqldb.org Il sito di riferimento del progetto HSQLDB**

IL PATTERN DAO

Finora abbiamo imparato ad utilizzare il framework *SQLMaps* contenuto nelle API di iBatis, infatti abbiamo visto come interagire con un DB usufruendo di pochi file XML e di un oggetto *SqlMapClient.* A livello di design però non siamo stati impeccabili in quanto abbiamo definito un data layer troppo legato alla tacnologia iBatis. Questa affermazione sembra fuoriluogo all'interno di questo articolo fortemente basato sulla conoscenza di iBatis; però nello sviluppo di un qualsiasi layer software è buona (ottima) regola non legarsi troppo ad una particolare tecnologia. Cosa succederebbe se per motivi progettuali fosse necessario cambiare framework e passare ad un altro tool (Hibernate, EJB tanto per fare nomi)? La risposta è semplice: basta sostituire ogni occorrenza dell'oggetto *sqlMap* con l'oggetto utilizzato dalla nuova tecnologia adottata. Questa soluzione non è il massimo in quanto a manutenibilità e quindi gestione del ciclo di vita del software. A tal proposito la nomenclatura menziona diversi pattern combinati tra loro che disaccoppiano i layer di un'applicazione. Per non allontanarci troppo dall'obiettivo di questo articolo ne adottiamo solo uno che risolve in modo soddisfacente il nostro problema. Infatti introdurremo il pattern DAO pensato appositamente per disaccoppiare la tecnologia utilizzata nel data layer dai layer che lo utilizzano. Il concetto che sta dietro alla sua implementazione è molto semplice, un *Database Access Object* (DAO) deve occuparsi delle operazioni di persistenza inerenti un *Value Object*. In poche parole per ogni oggetto *XyzVO* va definito un oggetto *XyzDAO* che contiene tanti metodi per quante sono le azioni di persistenza pensate per quell'oggetto. In verità iBatis prevede delle API per la realizzazione del DAO pattern, ma per non mettere troppa carne al fuoco e per non definire un'implementazione DAO ancora dipendente da iBatis, abbiamo voluto realizzarne una del tutto neutra. L'implementazione delle classi *SistemazioneDAO* e *DipendenteDAO* è semplice ed è contenuta nel CD della rivista. È importante invece osservare nella classe *DAOTest1* come cambia l'interazione con il DB grazie al loro impiego.

```
SistemazioneVO sist1= initSis( "I0-01", 0,
                              SistemazioneVO.UFFICIO);
SistemazioneVO sist2= initSis( "I0-02", 0,
                              SistemazioneVO.UFFICIO);
SistemazioneDAO sDao=new SistemazioneDAO();
sDao.insSistemazione(sist1);
sDao.insSistemazione(sist2);
DipendenteDAO dDao= new DipendenteDAO();
DipendenteVO roby= initDip( "Roberto", "Sidoti",
                              DipendenteVO.IMPIEGATO);
roby.setSistemazione(sist1);
dDao.insDipendente(roby);
roby.setMansione(DipendenteVO.DIRIGENTE);
dDao.updateDipendente(roby);
```

Come possiamo vedere, si effettuano operazioni sul DB senza in nessun modo avere cognizione dell'utilizzo di iBatis o di qualsiasi altro framework per la persistenza dei dati.

CONCLUSIONI

In questo articolo abbiamo conosciuto le principali caratteristiche di iBatis, abbiamo visto come un tool molto semplice riesca a sfruttare al meglio la duttilità dell'XML. Tale connubbio serve per la realizzazione di uno strato software che si occupa dell'integrazione di un RDBMS con un linguagio ad oggetti come Java. L'approccio adottato da iBatis si pone a metà strada tra l'impiego diretto delle API JDBC e l'utilizzo di tool molto potenti ma più complessi come Hibernate e gli EJB Entity. A supporto di questa trattazione abbiamo visto come pattern come il Value Object ed il DAO riescano a rendere le applicazioni snelle, flessibili e facilmente manutenibili.

Roberto Sidoti

L'AUTORE

Roberto Sidoti è ingegnere informatico, attaulmente si occupa di servizi VoIP basati sul protocollo SIP presso Telecom Italia LAB come consulente esterno. I suoi maggiori interessi riguardano le applicazioni distribuite sviluppate con tecnologia J2EE, da marzo 2005 si occupa dello sviluppo e della gestione del progetto GeiNuke
www.hostingjava.it/-geinuke/.

NOTA

INSTALLARE HSQL DB

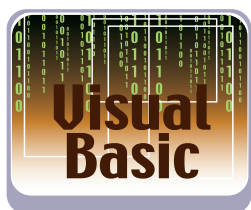
La versione 1.7.3 è reperibile all'indirizzo: http://sourceforge.net/project/showfiles.php?group_id=23316. **Effettuato il download del file zippato, bisogna scompattarlo in locale,A questo punto basta digitare da riga di comando**

```
>cd /hsqldb/bin
>runUtil
DatabaseManagerSwing
```

vedremo apparire un'applicazione grafica per la gestione dell'RDBMS.

PDF? Facciamoli con Visual Basic

Le principali caratteristiche del formato e le trasformazioni di un testo in documento PDF, utilizzando soltanto gli elementi primitivi di Visual Basic

CD WEB
VBePDF.zip
cdrom.ioprogrammo.it

REQUISITI

Conoscenze richieste
Conoscenze di base sulla gestione dei file e delle stringhe

Software
Piattaforma Windows 95 o superiore - Visual Basic 6 SP6

Impegno

Tempo di realizzazione

Il PDF – *Portable Document Format* – è una specifica pubblica per la creazione, lo scambio e la distribuzione di documenti elettronici affidabili e indipendenti dal dispositivo di visualizzazione. In altre parole, un documento sviluppato secondo le specifiche PDF preserva l'aspetto originale dei dati e può essere visualizzato con Windows piuttosto che con Linux, su un desktop piuttosto che su un Palmare. La versione 1.0 del PDF, chiamata *Carousel*, venne introdotta nel lontano 1993 poi nel 1996, con l'arrivo di Acrobat 2.0, seguì la versione 1.1 … fino all'attuale versione PDF 1.6 con Adobe Reader 7. Un documento in formato PDF può essere creato o modificato con vari strumenti di alto livello, l'ideale è Adobe Acrobat Writer che tra l'altro è perfettamente integrato con le applicazioni di MS Office. Dal punto di vista della programmazione il PDF è il diretto discendente dell'*Adobe PostScript* (PS), che è stato uno dei primi linguaggi per la descrizione di documenti. Per gestire e creare documenti PDF, a livello di programmazione, sono disponibili diverse tecniche tra le quali segnaliamo: quella basata sull'*FDF Toolkit for ActiveX*; quella che usa l'XSL-FO *(Formatting Objects)* e quella, utilizzata in questo articolo, che consiste nel creare un Documento PDF, in un file txt, utilizzando soltanto le specifiche PDF riportate nelle *PDF reference.*
Data l'importanza e la vastità dell'argomento, la trattazione sarà distribuita su due articoli: nel primo introdurremo le principali caratteristiche della specifica PDF; descriveremo come creare un semplice documento PDF ed implementeremo le routine Visual Basic che ne permettono la gestione. Nel successivo appuntamento, invece, amplieremo la trattazione sul PDF e tra l'altro presenteremo un'applicazione che permette di creare documenti PDF con i dati estratti da un database Access.

FORMATI APERTI- RTF, PS ECC

In generale si parla di testi non formattati (nel caso del TXT), testi formattati (HTML, RTF, DOC) e testi formattati per la stampa (PS, PDF). Il formato può essere di tipo proprietario o aperto, in quest'ultimo caso le specifiche di definizione sono di dominio pubblico. Tra i formati (o linguaggi di descrizione) pubblici ricordiamo il PDF, il PS e l'RTF. L'RTF è stato introdotto dalla Microsoft e per certi versi, ricalca le caratteristiche del formato DOC (però l'RTF non è adatto per la descrizione di formati non testuali).

IL FORMATO PDF

Un file PDF per l'utente finale è un documento di alto livello suddiviso in pagine, ognuna delle quali può contenere testo, grafici, elementi multimediali, links, ecc; e sulle quali è possibile effettuare ricerche testuali, impostare bookmark, annotazioni ed utilizzare elementi interattivi.

Document catalog — Page tree — Page (Content stream, Thumbnail image, Annotations) … Page; Outline hierarchy — Outline entry … Outline entry

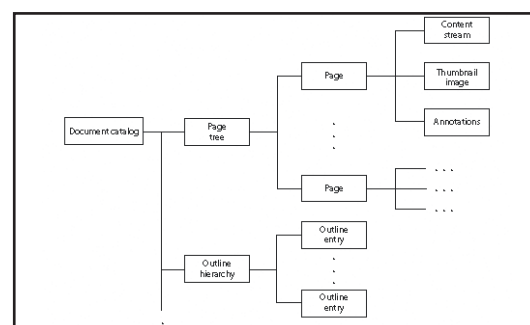



Fig. 1: **La struttura di un documento PDF**

A basso livello, o meglio a livello di programmazione, un semplice, documento PDF può essere visto come un insieme gerarchico di oggetti strutturati ad albero; dove (come vedremo più avanti) la radice è un oggetto di tipo *Catalog*, collegato ad una collezione di *Pages* (Pagine). Le *Pages* a loro volta contengono oggetti di tipo immagine, testo, font, grafici ecc. In altre parole ogni elemento mostrato su un documento PDF, a livello di programmazione, è un oggetto opportunamente inserito nel contesto, in modo

tale che sia facilmente individuabile per le successive rappresentazioni e ricerche. S'intuisce che un documento PDF, mantiene il "look and feel" originario proprio perché al suo interno incorpora tutti gli elementi necessari per la visualizzazione cioè font, immagini, grafici ecc. Iniziamo la panoramica sul PDF presentando i principali costrutti e la struttura base di un documento. Per maggiori dettagli sugli argomenti introdotti, v'invitiamo a consultare la *PDF Reference* relativa alla versione 1.6 scaricabile dal seguente URL: *http://partners.adobe.com/public/developer/pdf/topic.html.*

OGGETTI E TIPI DI DATI

In questo e nel successivo paragrafo presentiamo gli elementi PDF che permettono di creare un documento con testo colorato. In particolare, per ogni elemento PDF daremo una breve descrizione e faremo un esempio. In seguito, questi esempi, li raggrupperemo in un file testo per produrre un documento PDF che visualizzi il testo *"ioProgrammo"*, come mostrato in **Figura 2**.

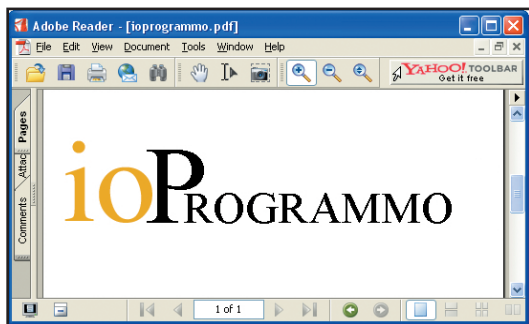



Fig. 2: Il file PDF con caratteri colorati

Il PDF supporta otto tipi di oggetti: numeric (integer e real number), name, boolean, string, array, dictionary, stream, null object. Questi, per essere referenziati, possono avere associata una label numerica. Gli oggetti con label sono nominati oggetti indiretti. La label, che permette l'identificazione univoca, è costituita da due numeri: uno sequenziale e l'altro di revisione; quest'ultimo, nei nostri esempi, è posto uguale a zero. Gli oggetti indiretti, in particolare, sono preceduti dalla label numerica e sono racchiusi tra le parole chiave *obj* e *endobj*. Di seguito riportiamo la struttura dell'oggetto *1 0*.

```
1 0 obj
 …
endobj
```

Quando in un documento si fa riferimento ad un oggetto indiretto si deve usare la parola chiave *Parent* seguita dai numeri che l'identificano e dalla lettera *"R"*. Dato che il PDF è case sensitive, attenzione al fatto che *"R"* è diversa da *"r"*. Un oggetto che ha il 3 0 come *parent* è il seguente.

```
2 0 obj
 /Parent 3 0 R
 …
endobj
```

Un oggetto dictionary, invece, è un insieme di coppie chiave/valore racchiusi tra i simboli << >>. Con esso grazie alla chiave */Type* si possono definire altri oggetti quali *Page, Pages, Catalog, Font* ecc. Di seguito è presentato un esempio di oggetto dictionary che fornisce informazioni sul documento che lo contiene. Queste informazioni sono visualizzate da Acrobat Reader nella finestra proprietà del documento (**Figura 3**). Inoltre, facciamo notare come chiariremo più avanti, che questo oggetto è nominato *Info.*

```
1 0 obj
<<
/CreationDate (D:20051006213203+01'00')
/Producer (ioprogrammo)
/Author (massimo)
/Title (provapag1.pdf)
/Creator (blocconote)
>>
Endobj
```

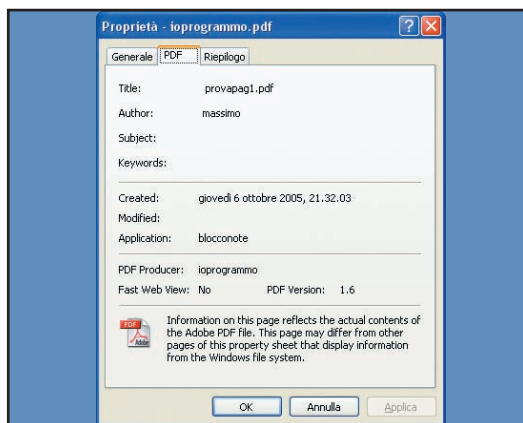



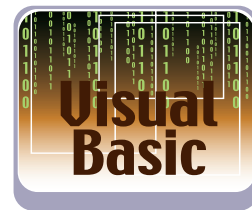
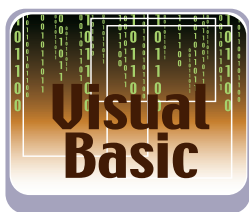
Fig. 3: Finestra proprietà di un file PDF

Nell'oggetto precedente il formato della data è del tipo *D:yyyymmddhhmmss +01'00'*, dove *+01'00'* fissa la differenza di orario rispetto a Greenwich (nel caso di ora legale è + 02). Ora presentiamo un esempio di oggetto *Page* cioè un oggetto *dictionary* con la chiave */Type /Page.*

```
4 0 obj
<<
/Type /Page
/Parent 5 0 R
/Resources
  << /Font << /F4 3 0 R >>
```

Visual Basic

I TUOI APPUNTI

Visual Basic

```
>>
/Contents 2 0 R
>>
endobj
```

Nell'oggetto 4 0 sono presenti alcuni riferimenti: alla pagina radice, con la chiave */Parent;* agli oggetti contenuti nella pagina con */Contents* e alle risorse di supporto a tali oggetti con la chiave */Resource*.
L'oggetto padre della pagina è il *5 0,* le risorse sono rappresentate da un Font i cui dettagli sono specificati nell'oggetto *3 0,* infine il contenuto della pagina è rappresentato dall'oggetto *2 0*. Di seguito riportiamo gli oggetti *2 0* e *3 0*.

```
2 0 obj
<<
/Length 121
>>
stream
BT
/F4 120 Tf
30 390 Td
0.9 0.7 0.02 rg
(IO) Tj
0 0 0 rg
/F4 150 Tf
105 0 Td
(P) Tj
/F4 60 Tf
50 0 Td
(ROGRAMMO) Tj
ET
endstream
endobj

3 0 obj
<< /Type /Font
/Subtype /Type1
/BaseFont /Times-Roman
>>
endobj
```

OPERATORE	DESCRIZIONE
Tc	**Imposta lo spazio tra i caratteri**
Tw	**Imposta lo spazio tra le parole**
Tz	**Imposta l'ampiezza orizzontale della riga di testo**
TL	**Imposta l'interlinea del testo**
Tf	**Imposta l'ampiezza e il nome del font**
Ts	**Stabilire l'elevazione del testo (cioè se è apice o pedice)**
TD	**Sposta il testo nella posizione specificata e setta l'interlinea**
TJ	**Visualizza il testo e permette d'impostare la posizione dei singoli caratteri**
Tj	**Mostra il testo**
T*	**Sposta il testo all'inizio di una nuova riga**
Tm	**Permette d'impostare (con una serie di valori) la matrice di disposizione del testo**
rg	**Utilizzato per impostare i colori RGB del testo**
BT, ET	**Segnalano l'inizio e la fine del testo**

Tabella 1: Alcuni operatori per la formattazione del testo.

Nell'oggetto *2 0* usiamo uno stream che definisce una sequenza di caratteri con operatori di formattazione. L'oggetto indiretto che contiene lo stream, con la chiave */Length* deve specificare il numero di byte comprese tra le parole stream e endstream. Il testo con gli operatori di formattazione, invece, deve essere preceduto e seguito dagli operatori *BT* (inizio testo) e *ET* (fine testo). Il testo *"IOPROGRAMMO"* (spezzato in *IO P ROGRAMMO*), è formattato con gli operatori: *Tf, Td, rg* e *Tj*. Con *Tf* si specifica il nome del Font da utilizzare e la dimensione del carattere. *Td* imposta la posizione dove scrivere i caratteri. Dato che le dimensioni della pagina sono in punti (come spieghiamo in seguito) con 30 390 *Td,* posizioniamo il testo (con carattere di grosse dimensioni) quasi al centro del foglio. L'operatore *rg,* invece, colora il testo in base ai tre colori fondamentali Rosso, Giallo e Blu. Facciamo notare che in PDF i valori dei colori sono compresi tra 0 e 1 e non tra 0 e 255 come in Visual Basic. Infine l'operatore *Tj* visualizza il testo compreso tra le parentesi tonde. L'oggetto *3 0* definisce il Font. Per ulteriori spiegazioni sui Font vi rimandiamo al successivo appuntamento e al box laterale. Ora vediamo come si definisce un oggetto *Pages* (che è il *parent* degli oggetti *Page*).

```
5 0 obj
<<
/Type /Pages
/Kids [4 0 R]
/Count 1
/MediaBox [0 0 612 792]
>>
endobj
```

Nell'oggetto *Pages* con la chiave */MediaBox,* e le coordinate di due punti, si specificano le dimensioni delle pagine. Nel nostro esempio la dimensione è l'A4, che in centimetri è 21 per 29,7; in pollici è 8.5 per 11, mentre in punti, l'unità di misura di default del PDF, è 612 per 792 punti (dato che un punto è 1/72 di pollici). Inoltre, facciamo notare, che in PDF il punto di coordinate (0,0) è in basso sul lato sinistro del foglio. Con la chiave */Count* s'indicano il numero di pagine figlie; mentre con */Kids* si specificano le loro label. Nel nostro caso, dato che si tratta di un documento con una pagina, c'è solo l'oggetto *4 0.* L'ultimo oggetto che consideriamo è il *catalog* definito con la chiave */Type /Catalog.* Questo oggetto di solito è la radice *(Root)* del documento e contiene un riferimento all'oggetto *Pages,* come mostrato di seguito.

```
6 0 obj
<<
/Type /Catalog
/Pages 5 0 R
>>
```

STRUTTURA DI UN FILE PDF

Un file PDF standard, e senza revisioni, è organizzato in 4 sezioni: *header, body, xref Table* e *Trailer.* L'*header*, o l'intestazione, è la prima riga del file e specifica la versione di PDF utilizzata. Per esempio se in un file la prima riga è la seguente *%PDF-1.5,* il file potrà essere letto solo con Acrobat Reader 6 (5+1) o versioni superiori.

Header

Body

Cross-reference table

Trailer

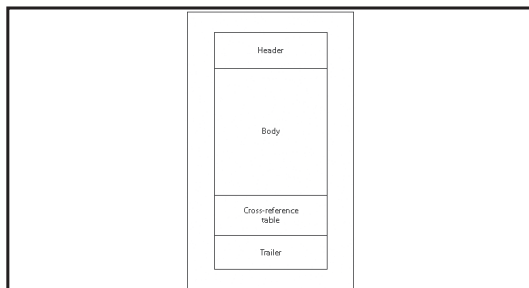



Fig. 4: **Le sezioni di un file PDF**

Il *Body*, o corpo, contiene la definizione degli oggetti indiretti che formano il documento, cioè degli oggetti simili a quelli illustrati nei precedenti paragrafi. Nella *Cross-Reference Table,* invece, sono specificati le posizioni – *offset* - in numero di byte (caratteri se consideriamo un file di testo) degli oggetti dichiarati nel *body;* c'è una riga di tabella per ogni oggetto. Queste informazioni sono utilizzate, in fase di ricerca, per l'accesso casuale agli elementi del documento (in realtà il documento è utilizzabile, anche se questi valori sono sbagliati, però ne risente la velocità di ricerca!). Ogni riga della tabella è lunga 20 byte. I primi 10 byte specificano il numero di carattere di offset, segue un spazio e 5 caratteri che danno il numero di generazione dell'oggetto (nel nostro caso sono posti a zero), segue un altro spazio e la lettera *"f"* o *"n"* che indicano se l'oggetto è libero *(free)* o in uso. Per il file del nostro esempio, la *Cross-Reference Table* è la seguente.

```
xref
0 7
0000000000 65535 f
0000000009 00000 n
0000000160 00000 n
0000000155 00000 n
0000000401 00000 n
0000000506 00000 n
0000000587 00000 n
```

Dopo *xref* è presenta 0 7 cioè il numero delle righe della tabella. Nel nostro caso il numero di righe è 7, anche se il documento presenta soltanto 6 oggetti, dato che la riga con offset zero (oggetto 0 0) è obbligatoria e serve per future revisioni del documento. L'ultima sezione di un file PDF è la *Trailer,* questa tra l'altro riporta l'offset della *Cross-Reference Table* (nell'esempio di sotto è 636) che ne permette una rapida individuazione in fase di ricerca. Inoltre, sono presenti i riferimenti ad alcuni oggetti speciali come la *Root* e l'*Info.* L'ultima riga della *Trailer,* e quindi del file, deve essere *%%EOF.* Ecco come si presenta la *Trailer* del nostro esempio.

```
trailer
<<
/Size 7
/Root 6 0 R
/Info 1 0 R
>>
startxref
636
%%EOF
```

Facciamo notare che la *Trailer* non è altro che un oggetto dictionary con le seguenti chiavi: */Size, /Root* e */Info.* Con *Size* si specifica il numero di sezioni (o punti d'ingresso) presenti nella tabella *xref; Root* dà un riferimento all'oggetto *catalog* presente nel file, *Info* contiene un riferimento all'oggetto che fornisce informazioni sul file (specificato in precedenza).

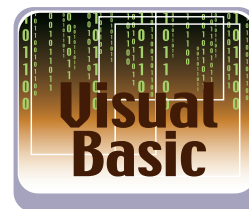
IL PRIMO PDF

Sulla base degli esempi precedenti possiamo creare il primo esempio di documento PDF. Questo l'otteniamo disponendo, opportunamente, in un file *txt* gli esempi precedenti e salvando il tutto con estensione *pdf.* Il documento che si ottiene lo trovate nel CD allegato alla rivista, in esso sono disponibili anche le procedure Visual Basic per la creazione automatica.

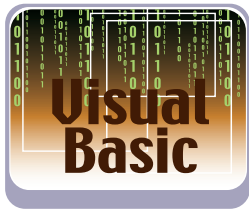
PDF CON VISUAL BASIC

In questo paragrafo presentiamo un progetto Visual Basic che ricava un documento PDF dal contenuto di un TextBox multilinea. Considerando gli esempi presentati negli altri paragrafi, in Visual Basic ci limiteremo a costruire un documento PDF di una pagina che contiene un solo oggetto stream di caratteri che usa il Font nominato *F4.* Per questo le quattro sezioni del documento sono fisse, mentre variano i caratteri inseriti nello stream e i valori degli offset degli oggetti. S'intuisce che il testo, dello stream, è quello inserito nel TextBox della form e che nel progetto abbiamo bisogno di una procedura che scrive, su un file *txt*, i dati in formato PDF.

1 Create un nuovo progetto e sulla form disponete tre textbox e un pulsante. I textbox servono a

Visual Basic

NOTE

COME CREARE UN PDF

I documenti PDF possono essere creati con diverse applicazioni (oltre che con Adobe Writer). Per esempio c'è la creazione OnLine, offerta dal sito http://createpdf.adobe.com/; ci sono tool Freeware ed altri a pagamento. Un tool FreeWare è PDF4Free, reperibile al link: http://www.pdfpdf.com/pdf4free.exe, esso installa un Driver per stampante in formato PDF. In questo caso per creare un documento, da un'applicazione, basta fare un'operazione di stampa. Se invece si deve convertire un documento PDF in Word si può utilizzare Scansoft PDF Converter, per maggiori informazioni collegatevi al sito www.scansoft.com/pdfconverter/.

specificare il testo da inserire nel documento (*TxtStrPdf* - multilinea), la dimensione del carattere (*TxtDimCar*) e il nome del file PDF (*TxtNomeFile*).

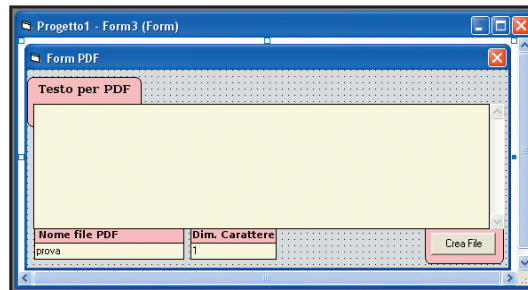



Fig. 5: **La form in fase di progettazione**

2 In un modulo *Bas* inserite le seguenti dichiarazioni e la procedura che scrive una riga su un file.

```
Public Const Numerofile = 1
Public NomeFile As String
Public Const VersionePDF = "%PDF-1.6"
Public Sub ScriviPDF(StrPdf As String)
  Print #Numerofile, StrPdf & vbCrLf;
  Debug.Print StrPdf
End Sub
```

Tra le dichiarazioni c'è la versione del PDF, il numero e il nome del file. Notate che la ScriviPDF scrive sul file e sulla finestra Immediata dell'IDE di Visual Basic.

SDK- FDF TOOLKIT FOR ACTIVEX

Per gestire a livello di programmazione il formato PDF è disponibile la libreria AcroExch che permette di utilizzare la maggior parte delle caratteristiche di Acrobat Reader. AcroExch contiene vari oggetti tra i quali App che permette di collegarsi ad una sessione di Acrobat Reader; AvDoc che controlla le proprietà del documento visualizzato; AvPageView che è la pagina visualizzata ecc. Nel successivo appuntamento descriveremo le principali caratteristiche di questo oggetto che per funzionare correttamente bisogna di Acrobat Writer. Per costruire un documento PDF, in base a dei template, invece, si può utilizzare l'FDF Toolkit (FDF-Form Data Format). Con questo Toolkit tra l'altro si possono creare degli interessanti report di stampa anche grafici. L'FDF Toolkit può essere scaricato pagando dal seguente link: http://partners.adobe.com/asn/developer/acrosdk/forms.html.

3 La procedura principale, cioè la *CreaFile*, è descritta sotto. Essa oltre alle istruzioni di apertura (*Open*) e chiusura (*Close*) del file contiene una serie di invocazione alla funzione *ScriviPDF* che consentono di costruire la struttura del file PDF.

```
Private Sub CreaFile_Click()
 Dim LungStream As Integer
 Dim objnum As Integer
 Dim strinstream As String
Dim lenstream As Integer
NomeFile = App.Path + "\" _
& TxtNomeFile + ".pdf"
Open NomeFile For Output As #Numerofile
 ScriviPDF VersionePDF
 objnum = 1
 ScriviPDF CStr(objnum) & " 0 obj"
 ScriviPDF "<<"
 ScriviPDF "/CreationDate " _
 & "(D:" & Format(Now(), "yyyymmddhhmmss")
                                      & "+01'00')"
 ' ORA legale +02
 ScriviPDF "/Producer (ioprogrammo)"
 ScriviPDF "/Author (massimo)"
 ScriviPDF "/Title ( provapag1.pdf )"
 ScriviPDF "/Creator  (blocconote)"
 ScriviPDF ">>"
 ScriviPDF "endobj"
 objnum = objnum + 1
 ScriviPDF CStr(objnum) & " 0 obj"
 strinstream = formattatesto
 lenstream = Len(strinstream) + 44
 '44 per gli altri caratteri di formattazione
 ScriviPDF "<<"
 ScriviPDF "/Length " + CStr(lenstream)
 ScriviPDF ">>"
 ScriviPDF "stream"
 ScriviPDF "BT"
 ScriviPDF "/F4 " + CStr(Me.Txtdimcar) + " Tf"
 ScriviPDF "12 0 0 12 20 750 Tm"
 ScriviPDF "0 -" + CStr(Me.Txtdimcar) + " TD"
 ScriviPDF strinstream
 ScriviPDF "ET"
 ScriviPDF "endstream"
 ScriviPDF "endobj"
 objnum = objnum + 1
 ScriviPDF CStr(objnum) & " 0 obj"
 ScriviPDF "<< /Type /Font"
 ScriviPDF "/Subtype /Type1"
 ScriviPDF "/BaseFont /Times-Roman"
 ScriviPDF ">>"
 ScriviPDF "endobj"
 objnum = objnum + 1
 ScriviPDF CStr(objnum) & " 0 obj"
 ScriviPDF "<<"
 ScriviPDF "/Type /Page"
 ScriviPDF "/Parent 5 0 R"
 ScriviPDF "/Resources"
 ScriviPDF "   << /Font << /F4 3 0 R >>"
 ScriviPDF ">>"
 ScriviPDF "/Contents 2 0 R"
 ScriviPDF ">>"
 ScriviPDF "endobj"
 objnum = objnum + 1
 ScriviPDF CStr(objnum) & " 0 obj"
 ScriviPDF "<<"
 ScriviPDF "/Type /Pages"
 ScriviPDF "/Kids [4 0 R]"
```

```
ScriviPDF "/Count 1"
ScriviPDF "/MediaBox [0 0 612 792]"
ScriviPDF ">>"
ScriviPDF "endobj"
objnum = objnum + 1
ScriviPDF CStr(objnum) & " 0 obj"
ScriviPDF "<<"
ScriviPDF "/Type /Catalog"
ScriviPDF "/Pages 5 0 R"
ScriviPDF ">>"
ScriviPDF "endobj"
ScriviPDF "xref"
ScriviPDF "0 7"
ScriviPDF "0000000000 65535 f"
ScriviPDF "0000000009 00000 n"
ScriviPDF Format(431 + lenstream, _
"0000000000") + " 00000 n"
ScriviPDF Format(148 + lenstream, _
"0000000000") + " 00000 n"
ScriviPDF Format(319 + lenstream, _
"0000000000") + " 00000 n"
ScriviPDF Format(501 + lenstream, _
"0000000000") + " 00000 n"
ScriviPDF Format(582 + lenstream, _
"0000000000") + " 00000 n"
ScriviPDF "trailer"
ScriviPDF "<<"
ScriviPDF "/Size 7"
ScriviPDF "/Root 6 0 R"
ScriviPDF "/Info 1 0 R"
ScriviPDF ">>"
ScriviPDF "startxref"
ScriviPDF CStr(631 + lenstream)
ScriviPDF "%%EOF"
Close #Numerofile
Dim Comodo As Double
Comodo = Shell("C:\Programmi\Adobe\Acrobat " _
& "7.0\Reader\AcroRd32.exe " + NomeFile)
End Sub
```

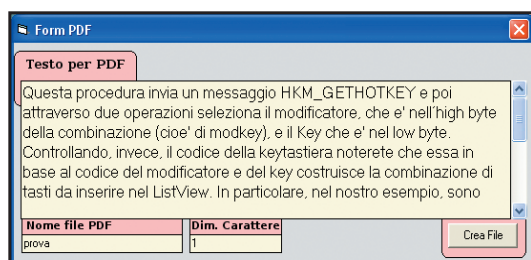



Fig. 6: La form che crea un file PDF con il contenuto di un TextBox

Facciamo notare che, nella procedura, gli offset e la lunghezza dello stream sono valutati in base al numero di caratteri inseriti nel textbox.

4 Nella procedura *CreaFile* è invocata la formattatesto che formatta il testo, inserito nel textbox, con gli operatori *Tj* e *T** e le parentesi tonde.

```
Private Function formattatesto() As String
 Dim com1, com2, wstr, _
 repstrin, parti As String
 Dim riga, j, pos As Integer
 riga = Int(100 / Me.Txtdimcar)
 wstr = Trim(TxtStrPdf.Text)
 wstr = Replace(wstr, vbCr, " ")
 wstr = Replace(wstr, vbCrLf, " ")
 wstr = Replace(wstr, vbLf, " ")
 If Len(wstr) > riga Then
  parti = Int(Len(wstr) / riga)
  For j = 1 To parti
   pos = j * riga
    While Asc((Mid(wstr, pos, 1))) <> Asc(" ")
     pos = pos - 1
    Wend
   com1 = Mid(wstr, 1, pos - 1)
   com2 = Mid(wstr, pos + 1, Len(wstr) - pos)
   wstr = com1 + vbCr + com2
  Next j
 End If
 wstr = " (" + wstr + vbCr
 repstrin = ") Tj T* ("
 wstr = Replace(wstr, vbCr, repstrin)
 wstr = Left(wstr, Len(wstr) - 5)
 formattatesto = wstr
End Function
```

Nella procedura formattatesto, per semplificare, dividiamo la stringa di input in righe di massimo 100 caratteri (o sottomultipli, in base alla dimensione del carattere) e togliamo i "ritorno a capo" originali. Inoltre, quando dividiamo il testo in righe inseriamo i "ritorno a capo" tra due parole.
V'invitiamo a rendere la procedura precedente più efficiente e funzionale.

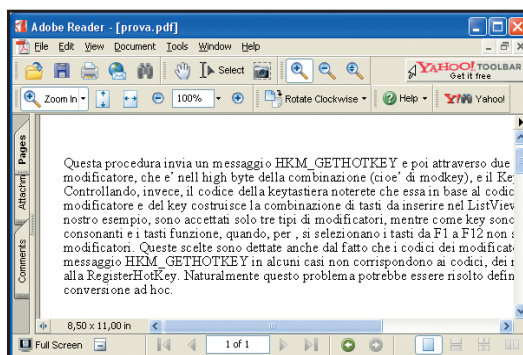



Fig. 7: Il File PDF con il contenuto del TextBox

CONCLUSIONI

In questo articolo abbiamo descritto come creare un documento PDF che contiene del testo. Nel successivo appuntamento presenteremo un'applicazione che crea dei report PDF anche grafici.

Massimo Autiero

NOTE

FONT

Nel PDF sono disponibili 14 Font predefiniti, cioè Font che per essere utilizzati non hanno bisogno di particolari accorgimenti. Questi Font, classificati come Type1 sono:

CourierNew *(Italic, Bold, BoldItalic)*, Arial, TimesNewRoman, ZapfDingbats (solo un tipo) e Symbol (solo un tipo).

Negli esempi abbiamo visto che per utilizzare un *Font Type1*, bisogna definirlo in un oggetto e specificarlo, come risorsa (*Resources*) della pagina con la chiave */Font*.

XSL, usare cicli e condizioni

In quest'articolo illustreremo la sintassi e la semantica dei costrutti messi a disposizione da XSL per controllare il flusso d'esecuzione di una trasformazione. Concluderemo con un esempio in VB.NET

❑ CD ☑ WEB
Xsl.zip
cdrom.ioprogrammo.it

REQUISITI
Conoscenze richieste
Basi di programmazione Java
Software
JDK 1.4 o superiore
Impegno
Tempo di realizzazione

Ogni linguaggio di programmazione non può prescindere da costrutti che consentano di definire condizioni. Anche XSL (pur non essendo ripetiamo un vero e proprio linguaggio di programmazione ma piuttosto un insieme di regole dichiarative destinate ad essere interpretate) non sfugge alla regola. Gli elementi che esprimono le istruzioni condizionali sono:

1. *<xsl:if>*
2. *<xsl:choose>*

IL CONTROLLORE DI FLUSSI "IF"

Il primo elemento, *<xsl:if>* è un semplice if ad una sola condizione. L'espressione da verificare è contenuta dell'attributo test , ad esempio:

```
<xsl:if test="name='rossi' ">
Il nome è Rossi
</xsl:if>
```

L'attributo test contiene la condizione da verificare nel documento di origine. Se l'espressione in questo attributo è vera il contenuto di *<xsl:if>* è inserito nell'output. L'espressione valutata dall'attributo *test* può essere un Nodo del documento xml nel contesto corrente o una comparazione tra stringhe o numeri, da notare che le stringhe all'interno delle espressioni si scrivono tra gli apici singoli; il valore *"rossi"* è una stringa mentre senza apici rossi sarebbe il riferimento al nodo rossi del documento di origine. I numeri invece si esprimono senza apici. Sugli operatori di comparazione da usare nelle espressioni condizionali vedi il box allegato. L'elemento *<xsl:if>* rappresenta una condizione semplice: se la condizione è vera viene prodotto un certo output. Ma che succede se dobbiamo verificare più condizioni alternative? In Xsl non esiste un elemento *"else"* o *"elseif"* interno ad *<xsl:if>*, dobbiamo far riferimento ad un costrutto più flessibile quale appunto *<xsl:choose>*.

IL CONTROLLORE DI FLUSSO "CHOOSE"

<xsl:choose> è concettualmente simile al *Select* di visual basic e si compone di due sotto-elementi :

- **<xsl:when>** che contiene l'attributo *"test"* che rappresenta la condizione da verificare.
- **<xsl:otherwise>** che rappresenta il caso in cui nessuna delle condizioni espresse dagli elementi *<xsl:when>* si sia verificata.

Com'è facilmente intuibile *<xsl:choose>* deve contenere almeno un sotto-elemento *<xsl:when>* (altrimenti non ci sarebbe nessuna condizione da verificare). Com'è altrettanto ovvio che vi possa essere soltanto un sotto elemento *<xsl:otherwise>* mentre non vi sono limiti ai *<xsl:when>* da utilizzare.
Ma passiamo subito ad un esempio di *<xsl:choose>* :

```
<xsl:choose>
```

PUNTATE PRECEDENTI

Possiamo definire XSL come: un insieme di istruzioni per trasformare un input, in formato XML, in vari tipi di output (altro XML, HTML, testo ecc...).Le istruzioni che occorrono per dare il via alla trasformazioni sono anch'esse scritte in formato XML. XSL infatti non è altro che una applicazione specializzata di XML, nata con lo scopo di gestire la trasformazione dei dati.XSL si basa sulla lettura dei dati da una fonte XML, e legge i contenuti dal documento di origine attraverso un linguaggio di selezione dei nodi XML chiamato XPath. Possiamo pensare a XPath come all'equivalente di SQL per un database, cioè ad una particolare sintassi per estrarre le informazioni da una fonte dei dati.
XPath non fa parte in senso stretto di XSL anche perché viene integrato spesso anche in altre tecnologie come i parser basati su DOM. Tuttavia rimane ed è alla base del meccanismo di trasformazione.

```
<xsl:when test="name='Antonio'">
   Benvenuto Antonio
</xsl:when>
<xsl:when test="name='Giovanni'">
   Benvenuto Giovanni
</xsl:when>
<xsl:otherwise>
   Benvenuto chiunque tu sia
</xsl:otherwise>
</xsl:choose>
```

Nella prima condizione viene verificata la corrispondenza che contenuto del nodo *<name>* al valore letterale *"Antonio"* e prodotto un certo output; nella seconda si verifica la corrispondenza rispetto al valore *"Giovanni"* ed infine si fornisce una soluzione di Default.

I CICLI

Nei linguaggi di programmazione classici si intende per ciclo un costrutto che consenta di ripetere una certa operazione più volte fino a che non si verifica una data condizione. Non diverso è il concetto di ciclo in XSL dove è rappresentato dall'elemento *<xsl:for-each>*. I cicli con *<xsl:for-each>* consentono di ripetere un determinato output per tutti i nodi individuati utilizzando l'attributo *"select"*. Prendiamo ad esempio questo frammento di codice:

```
<table>
<xsl:for-each select="indirizzi">
  <xsl:for-each select="indirizzo">
     <tr>
        <td>
     <xsl:value-of select="nome"/>
        </td>
        <td>
     <xsl:value-of select="cognome"/>
        </td>
           <td>
     <xsl:value-of select="via"/>
           </td>
           <td>
     <xsl:value-of select="comune"/>
           </td>
        </tr>
     </xsl:for-each>
  </xsl:for-each>
</table>
```

Qui abbiamo due cicli nidificati, il primo legge tutti i nodi *<indirizzi>* del documento XML, quando trova un nodo passa il controllo al secondo ciclo che, all'interno del nodo *<indirizzi>*, legge tutti i sottonodi *<indirizzo>*. Il secondo ciclo può quindi accedere ai sotto-nodi di *<indirizzo>* che vengono inseriti all'interno di un output HTML standard. È importante sottolineare come all'interno di un ciclo *<xsl:for-each>* cambia il contesto di lettura, cioè come tutti i riferimenti a nodi siano relativi al nodo correntemente esaminato, per cui il riferimento al nodo *nome* nel contesto del secondo ciclo del nostro esempio sarà corrispondente a *indirizzi/indirizzo/nome.* Naturalmente nell'output ci può essere bisogno di far riferimento a un nodo posto al di fuori del contesto corrente.
Consideriamo questo documento XML:

```
<province>
        <provincia>
                <sigla>MI</sigla>
                <comune>
<nome>Milano</nome>
                </comune>
        </provincia>
        <provincia>
                <sigla>ROMA</sigla>
                <comune>
                        <nome>Roma</nome>
                </comune>
        </provincia>
</province>
```

Poniamo che da questo documento dovessimo ricavare una semplice tabella HTML che mostri il nome del comune con accanto la sigla della sua provincia. Esprimendo il tutto con una serie di *<xsl:for-each>*

HELLO WORLD XSL

Potevamo mancare all'appuntamento con il famoso "*Hello World*"? Ovviamente no! Vediamo quindi un semplice esempio di XSL in azione. Partiamo da un semplice file XML

```
<?xml version="1.0"?>
<helloworld>
  <titolo>Hello world</titolo>
  <messaggio>Benvenuti al corso XSL
        di ioProgrammo!</messaggio>
</helloworld>
```

Nella stessa directory poi creiamo il file XSL (nel CD *helloworld.xsl*)

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:
      xsl="http://www.w3.org/1999/XSL
                          /Transform">
<xsl:template match="/ ">
<html>
      <head>
      <title>Esempio Hello
World</title>
      </head>
      <body>
  <h1><xsl:value-of select=
                 "helloworld/titolo"/></h1>
        <div>
            <i>
          <xsl:value-of select=
                 "helloworld/messaggio"/>
            </i>
        </div>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Adesso modifichiamo il file XML inserendo il riferimento al foglio di stile nel modo seguente:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href=
                    "helloworld.xsl"?>
<helloworld>
  <titolo>Hello world</titolo>
  <messaggio>Benvenuti al corso XSL
       di IOProgrammo!</messaggio>
</helloworld>
```

Se riapriamo adesso il file XML nel browser, visualizzeremo la trasformazione definita nel file XSL.

potremmo scrivere:

```
<table>
  <xsl:for-each select="province">
    <xsl:for-each select="provincia">
      <xsl:for-each select="comune">
        <tr>
          <td>
            <xsl:value-of select="nome"/>
          </td>
          <td>
            <xsl:value-of select="../sigla"/>
          </td>
        </tr>
      </xsl:for-each>
    </xsl:for-each>
  </xsl:for-each>
</table>
```

Notiamo che sotto il ciclo *"comune"* dobbiamo leggere il nodo *"sigla"* che si trova sotto il nodo superiore *<provincia>* e quindi non è figlio *(child)* di *<comune>* ma è come lui figlio di *<provincia>*. Qui abbiamo fatto ricorso dell'espressione *"../sigla"* per selezionare il nodo sigla a partire dal contesto relativo al nodo superiore a quello corrente. Questa sintassi di selezione del nodo è espressa in un linguaggio chiamato *XPATH*. *XPATH* offre una grande flessibilità per la selezione di nodi, attributi e quant'altro basti pensare a come sarebbe stato possibile esprimere l'esempio precedente utilizzando *XPATH* pienamente:

```
<table>
  <xsl:for-each select="province/provincia/comune">
    <tr>
      <td>
  <xsl:value-of select="nome"/>
    </td>
      <td>
    <xsl:value-of select="../sigla"/>
      </td>
        </tr>
   </xsl:for-each>
</table>
```

Risparmiamo cioè due cicli *<xsl:for-each>* semplificando di molto la sintassi e la leggibilità.

IF O CHOOSE ?

È preferibile utilizzare il costrutto *<xsl:if>* o *<xsl:choose>* per esprimere condizioni? *<xsl:if>* ha il vantaggio di essere più conciso e quindi rende il codice meno lungo e più facile da leggere, ma che succede se, dopo aver distribuito *<xsl:if>* a destra e a manca nel nostro codice, ci accorgiamo che le condizioni da valutare non sarebbero una ma due o più? Dovremmo a quel punto trasformare tutti gli *<xsl:if>* in *<xsl:choose>* il che, in alcuni casi, potrebbe essere un'operazione abbastanza lunga e soggetta ad errori. D'altra parte *<xsl:choose>* supporta anche una sola condizione (con un solo *<xsl:when>*) quindi, a meno di non essere assolutamente sicuri che la condizione da valutare non può essere che una, conviene sempre utilizzare l'elemento <xsl:choose>.

Da quanto sopra comprendiamo come sia ormai arrivato il momento di prendere in esame *XPATH!* Ed è quello che faremo dalla prossima puntata.

TRASFORMARE CON XSL: NON SOLO WEB!

Abbiamo detto nelle scorse puntate che uno degli scopi di questo corso è quello di evidenziare l'utilità di XSL in varie occasioni. Molti infatti si limitano ad utilizzare XSL come una specie di foglio di stile *CSS* per formattare l'HTML. Certo XSL è anche questo, ma non solo. Ma vediamo un esempio concreto scritto in VB.NET. Il problema è un classico: vogliamo esportare una tabella di database in formato testo riconoscibile e importabile con Excel. Impostiamo il programma come console application ed iniziamo con l'impostare una funzione che legge i dati dal database in un dataset:

```
Private Function GetDataset() As DataSet
  'imposta la connessione
  Dim connessione As New OleDbConnection("Data
                        Source=" & DBPath & ";Provider=
                              Microsoft.Jet.OLEDB.4.0;")
  Dim comando As New OleDbCommand("Select *
                              from Clienti", connessione)
  Dim adapter As New OleDbDataAdapter(comando)
  Dim ds As New DataSet("DSClienti") ' creo il
                                          dataset vuoto
  adapter.Fill(ds)
  Return ds
End Function
```

Fin qui niente di nuovo. Utilizziamo una connessione al database (qui abbiamo utilizzato il classico *NorthWind.mdb* che è il database di esempio di Access) per riempire un Dataset con un *DataAdapter.* Una volta che abbiamo un dataset è facile convertirlo in un oggetto XML interpretabile come una sorgente di dati dal motore XSL. Lo facciamo con questa funzione:

```
Private Function EstraiDatiXML()
                                  As XPath.XPathDocument
     Dim ds As DataSet = GetDataset()
     Dim mem As New MemoryStream
     ds.WriteXml(mem, XmlReadMode.Auto)
     mem.Seek(0, SeekOrigin.Begin) 'riporta il
                                          flusso all'inizio
     Dim doc As New XPath.XPathDocument(mem)
     mem.Close()
     Return doc
  End Function
```

Che semplicemente restituisce un oggetto *XPath-Document* (più veloce da caricare rispetto ad un

XMLDocument) scrivendo l'output del metodo *WriteXml* del Dataset in un flusso di memoria ed utilizzando quest'ultimo per creare un nuovo *XPathDocument*. A questo punto dovremmo avere in memoria un documento di questo tipo:

```
<DSClienti>
  <xs:schema id="DSClienti" xmlns="" xmlns:xs=
                "http://www.w3.org/2001/XMLSchema"
                xmlns:msdata="urn:schemas-microsoft
                                   -com:xml-msdata">
    <xs:element name="DSClienti" msdata
           :IsDataSet="true" msdata:Locale="it-IT">
      <xs:complexType>
        <xs:choice maxOccurs="unbounded">
          <xs:element name="Table">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="IDCliente" type=
                        "xs:string" minOccurs="0" />
                <xs:element name="NomeSocietà" type=
                        "xs:string" minOccurs="0" />
                <xs:element name="Contatto" type=
                        "xs:string" minOccurs="0" />
                <xs:element name="Posizione" type=
                        "xs:string" minOccurs="0" />
                <xs:element name="Indirizzo" type=
                        "xs:string" minOccurs="0" />
                <xs:element name="Città" type=
                        "xs:string" minOccurs="0" />
                <xs:element name="Zona" type=
                        "xs:string" minOccurs="0" />
                <xs:element name="CAP" type=
                        "xs:string" minOccurs="0" />
                <xs:element name="Paese" type=
                        "xs:string" minOccurs="0" />
                <xs:element name="Telefono" type=
                        "xs:string" minOccurs="0" />
                <xs:element name="Fax" type=
                        "xs:string" minOccurs="0" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:choice>
      </xs:complexType>
    </xs:element>
  </xs:schema>
  <Table>
    <IDCliente>ALFKI</IDCliente>
    <NomeSocietà>Alfreds Futterkiste</NomeSocietà>
    <Contatto>Maria Anders</Contatto>
    <Posizione>Rappresentante</Posizione>
    <Indirizzo>Obere Str. 57</Indirizzo>
    <Città>Berlino</Città>
    <CAP>12209</CAP>
    <Paese>Germania</Paese>
    <Telefono>030-0074321</Telefono>
    <Fax>030-0076545</Fax>
  </Table>
... le altre righe sono omesse
</DSClienti>
```

Un documento quindi che può essere facilmente trasformato con XSL. Prepariamo quindi un documento XSL che definisce le regole di trasformazione che chiameremo *"formattaTxt.xsl"* e mettiamo nella stessa directory del programma. Da notare che, poiché dovremmo far riferimento anche a nodi con il prefisso *xs:*, il relativo namespace dovrà essere dichiarato anche nel nodo *<xsl:stylesheet>:*

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

Nel documento XSL impostiamo poi anche il metodo per la trasformazione che sarà in questo caso text e due parametri che rappresentano rispettivamente il valore del qualificatore di testo (in questo caso usiamo le doppie virgolette ") ed il separatore (usiamo il punto e virgola ;).

```
<xsl:param name="text-qualifier">"</xsl:param>
<xsl:param name="separator">;</xsl:param>
<xsl:output method="text"/>
```

Definiamo quindi un template *"formatta-entry"* che possiamo richiamare ogni volta che c'è da scrivere un valore:

```
<xsl:template name="formatta-entry">
        <xsl:param name="nodo"/>
        <xsl:value-of select="$text-qualifier"/>
        <xsl:value-of select="normalize-space(
                                          $nodo)"/>
        <xsl:value-of select="$text-qualifier"/>
          <xsl:choose>
        <xsl:when test="position()!=last()">
        <xsl:value-of select="$separator"/>
                 </xsl:when>
          </xsl:choose>
</xsl:template>
```

Questo template accetta come parametro il valore di un nodo (privato degli spazi in eccesso con la funzione *normalize-space()* di *XPath*) che decora con il qualificatore di testo e, se il nodo non è l'ultimo del gruppo, inserisce in coda il separatore. Prepariamo anche un altro semplice template *"nuova-riga"* che scrive il carattere di fine riga espresso come entità *xml
 :*

```
<xsl:template name="nuova-riga">
                          &#10;</xsl:template>
```

Adesso non ci resta che definire due template che

SUL WEB

Specifiche del linguaggio XSLT versione 1.0 (in lingua inglese)
http://www.w3.org/TR/xslt

Tutorial XSL e XPath (in lingua inglese)
http://www.topxml.com/xsl/tutorials/intro/default.asp

XSL School su *w3schools* (in lingua inglese)
http://www.w3schools.com/xsl/xsl_languages.asp

trasformano i dati della sorgente XML andando a formare la prima riga e le righe successive:

```
<xsl:template name="prima-riga">
  <xsl:for-each select=
          "//xs:schema//xs:sequence/xs:element">
  <xsl:call-template name="formatta-entry">
  <xsl:with-param name="nodo" select="@name"/>
  </xsl:call-template>
  </xsl:for-each>
  <xsl:call-template name="nuova-riga">
                            </xsl:call-template>
</xsl:template>
<xsl:template name="altre-righe">
  <xsl:for-each select="//Table">
    <xsl:for-each select="*">
      <xsl:call-template name="formatta-entry">
        <xsl:with-param name="nodo" select="."/>
      </xsl:call-template>
    </xsl:for-each>
    <xsl:call-template name="nuova-riga">
                            </xsl:call-template>
  </xsl:for-each>
</xsl:template>
```

Richiamiamo i due template nel template di avvio ed il file XSL è completo:

```
<xsl:template match="/">
  <xsl:call-template name="prima-riga">
                            </xsl:call-template>
  <xsl:call-template name="altre-righe">
                            </xsl:call-template>
</xsl:template>
```

A questo punto, nel sorgente del nostro programma VB.NET non resta che scrivere la procedura che partendo dalla rappresentazione XML del dataset la trasforma utilizzando il documento XSL e scrive l'output in un file di testo:

```
Private Sub TrasformaDatiInTxt()
    Dim engine As New Xsl.XslTransform
    engine.Load("formattaTxt.xsl")
    Dim outWriter As StreamWriter
    outWriter = File.CreateText(OUTPath)
                  'OUTPath è il percorso del file .txt
    engine.Transform(EstraiDatiXML, Nothing,
                                outWriter, Nothing)
End Sub
```

Richiamiamo il tutto dal *Main()* del programma ed il gioco è fatto:

```
Sub Main()
    Try
        TrasformaDatiInTxt()
        Console.WriteLine("Operazione completata")
    Catch ex As Exception
        Console.WriteLine("Errori:")
        Console.WriteLine(ex.ToString)
    End Try
End Sub
```

Un'ultima annotazione: quando importiamo il risultato in Excel dovremo impostare UTF-8 come valore *"origine file"* affinché il testo venga correttamente riconosciuto perché questa è la codifica di default dell'output XSL. Il vantaggio allo stesso risultato ottenuto senza XSL è che così esternializziamo la logica di costruzione dell'output rispetto al programma. Quando dobbiamo cambiare qualcosa nell'output (un qualificatore di testo, un separatore diverso ecc...) non abbiamo bisogno di ricompilare basta apportare le modifiche al file XSL. Quindi XSL non solo per il Web ma anche come valida risorsa nella programmazione desktop!

Francesco Smelzo

GLI OPERATORI DI COMPARAZIONE NELLE ESPRESSIONI CONDIZIONALI

Gli operatori di comparazione sono:

"="	**Operatore di eguaglianza , uguale a . Applicabile a stringhe e a numeri.**
<xsl:if test= "name='rossi' ">	**Valuta se il testo contenuto nel nodo name è uguale alla stringa 'rossi'**
<xsl:if test= "id= 0 ">	**Valuta se il testo contenuto nel nodo id interpretato come numero è uguale al numero 0**
"!="	**Operatorie di disuguaglianza, diverso da. Applicabile a stringhe e a numeri**
<xsl:if test= "name!='rossi' ">	**Valuta se il testo contenuto nel nodo name è diverso dalla stringa *"rossi"***
<xsl:if test="id!= 0 ">	**Valuta se il testo contenuto nel nodo id interpretato come numero è diverso dal numero 0**

Vi sono poi anche operatori di comparazione che operano solo sui numeri è sono: > (maggiore) e < (minore). Tuttavia all'interno delle espressioni non possono essere indicati direttamente perché, secondo la sintassi XML > e < non possono essere usati all'interno degli attributi. Si ricorre allora alle corrispondenti entità: > per > e < per <. Quindi avremo:

">"	**Operatore di confronto "maggiore". Applicabile solo a numeri.**
<xsl:if test="id > 0 ">	**Valuta se il testo contenuto nel nodo id interpretato come numero è maggiore del numero 0**
"<"	**Operatore di confronto "minore". Applicabile solo a numeri.**
<xsl:if test="id < 10 ">	**Valuta se il testo contenuto nel nodo id interpretato come numero è minore del numero 10**

Naturalmente è possibile combinare gli operatori > e < con l'operatore = per ottenere un confronto maggiore/uguale a o minore/uguale a:

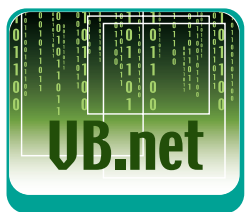
<xsl:if test="id < = 10 ">	**Valuta se il testo contenuto nel nodo id interpretato come numero è minore o uguale a 10**

L'inserimento nell'espressione di test del solo nome di un nodo, invece, produce il valore *True* se il nodo esiste e *False* in caso contrario:

<xsl:if test="name">	**Valuta se nel contesto corrente esistono uno o più nodi chiamati *"name"***

Fare Grafica con Visual Basic.NET

Continua il viaggio all'interno della tecnologia GDI+. In questo numero tratteremo il disegno di immagini e le problematiche legate al sistema di coordinate

VB.net

Nel primo articolo sulla tecnologia GDI+, ci siamo occupati del disegno di elementi grafici ed abbiamo visto come la prima operazione fondamentale, consiste nell'ottenere un riferimento ad un oggetto *Graphics*. Un oggetto Graphics rappresenta, in pratica, una superficie di disegno (non deve essere necessariamente una superficie visibile), normalmente l'area client di un oggetto *Form*. Fino ad ora, tutti gli esempi di codice assumevano che l'origine delle coordinate fosse l'angolo superiore sinistro dell'area client, e che tutti i valori fossero espressi in pixel. In questo articolo vedremo come, l'oggetto *Graphics* permetta di modificare le impostazioni predefinite cambiando il sistema di coordinate. La seconda parte dell'articolo sarà incentrata sulla gestione delle immagini di tipo raster e vettoriali.

I SISTEMI DI COORDINATE

In GDI+, quando si disegna un oggetto grafico (incluse immagini e testo), vengono utilizzati tre differenti sistemi di coordinate: di ambiente, di pagina e di periferica. In dettaglio:

- **Il sistema di coordinate di ambiente (o complessivo) è il sistema utilizzato di default per modellare l'ambiente grafico**. Tutte le coordinate che vengono passate ai metodi grafici sono espresse nel sistema di coordinate di ambiente.
- **Il sistema di coordinate della pagina è il sistema utilizzato dalla superficie su cui si disegna, sia essa una form, un controllo o un documento da stampare.** Di solito, l'origine e la scala di questo sistema coincidono con quelle del sistema di coordinate d'ambiente, ma ciò non è strettamente necessario. È possibile fissare l'origine del sistema in un punto qualsiasi della form. La conversione tra le coordinate dell'ambiente e quelle della pagina viene definita trasformazione d'ambiente.
- **Il sistema di coordinate della periferica è il sistema utilizzato dalla periferica fisica sulla quale avviene l'operazione di disegno.** Se si disegna su schermo la periferica è la finestra, se l'output viene inviato alla stampante la periferica diventa il foglio. È possibile definire con precisione l'unità di misura del sistema (pollici,millimetri, ecc), il rapporto tra l'asse verticale e quello orizzontale. La conversione tra le coordinate della pagina e quelle della periferica viene definita trasformazione di pagina.

REQUISITI

Conoscenze richieste
Conoscenze base di VB.NET

Software
Windows 2000/XP. Visual Basic .NET 2003

Impegno

Tempo di realizzazione

DISEGNARE UNA RETTA NELLO SPAZIO DI COORDINATE COMPLESSIVO

Per disegnare una linea retta, che colleghi due punti specificati da due coppie di coordinate, è sufficiente utilizzare il metodo *DrawLine* descritto nell'articolo precedente. Se, ad esempio, vogliamo disegnare una linea di colore verde che colleghi i punti di coordinate (1,1) e (150,150), possiamo farlo come segue. Si dichiara una variabile oggetto *OggettoGrafico* di tipo *Graphics* e si ottiene il riferimento alla superficie di disegno della form, utilizzando il metodo *CreateGraphics* esposto dalla form e da tutti i controlli di VB.Net.

```
Dim OggettoGrafico As Graphics = Me.CreateGraphics
```

Si dichiara una variabile oggetto *OggettoPen* di tipo Pen per creare una penna con un particolare colore (ad esempio Verde):

```
Dim OggettoPen = New Pen(Color.Green)
```

Si disegna la retta che collega i due punti specificati

Fig. 1: In figura il disegno della linea

dalle due coppie di coordinate, con il colore indicato dall'oggetto *Pen,* utilizzando il metodo *DrawLine OggettoGrafico.DrawLine(OggettoPen, 1, 1, 150, 150)* Si distrugge esplicitamente l'oggetto *Graphics* prima di uscire dalla procedura;

```
OggettoGrafico.Dispose()
```

Con l'istruzione *OggettoGrafico.DrawLine(OggettoPen, 1, 1, 150, 150),* i punti passati al metodo *DrawLine* si trovano nello spazio di coordinate complessivo, per cui il risultato è quello riportato in figura, in cui il punto di coordinate (1,1) coincide con l'angolo superiore sinistro della finestra

MODIFICARE IL SISTEMA DI COORDINATE

Descriviamo, ora, come fare riferimento ad un sistema di coordinate la cui origine si trovi in un punto all'interno dell'area client. Supponiamo, ad esempio, che l'origine si debba trovare a 40 pixel di distanza dal margine sinistro, ed a 30 pixel di distanza dall'estremità superiore. Tra i metodi che l'oggetto *Graphics* mette a disposizione, per influire sulla trasformazione d'ambiente, possiamo utilizzare il metodo *TranslateTransform.* Il metodo *TranslateTransform* permette di modificare l'origine del sistema di coordinate, spostandola nei punti specificati come argomento. In questo modo, per disegnare lo stesso oggetto in una posizione differente, sarà sufficiente traslare l'origine degli assi *X-Y* di 40 unità nella direzione *x* e 30 unità nella direzione *y.* Per disegnare la linea dell'esempio precedente, nel nuovo sistema di coordinate, possiamo scrivere:

```
Dim OggettoGrafico As Graphics = Me.CreateGraphics
Dim OggettoPen = New Pen(Color.Green)
OggettoGrafico.TranslateTransform(40, 30)
OggettoGrafico.DrawLine(OggettoPen, 1, 1, 150, 150)
OggettoGrafico.Dispose()
```

Fig. 2: In figura il disegno della linea nel nuovo sistema di coordinate

Un altro metodo, degno di attenzione, è il metodo *RotateTransform.* Il metodo *RotateTransform* permette di ruotare di un angolo qualsiasi, tutto ciò che viene disegnato sull'oggetto *Graphics.* Se, ad esempio, vogliamo disegnare un rettangolo, utilizzando il metodo *DrawRectangle,* ed applicargli una rotazione di 45 gradi, possiamo scrivere il seguente codice:

```
Dim OggettoGrafico As Graphics = Me.CreateGraphics
Dim OggettoPen = New Pen(Color.Blue)
'applico la rotazione di 45 gradi
OggettoGrafico.RotateTransform(45)
OggettoGrafico.DrawRectangle(OggettoPen, 100, 10,
                                              100, 50)
OggettoGrafico.Dispose()
```

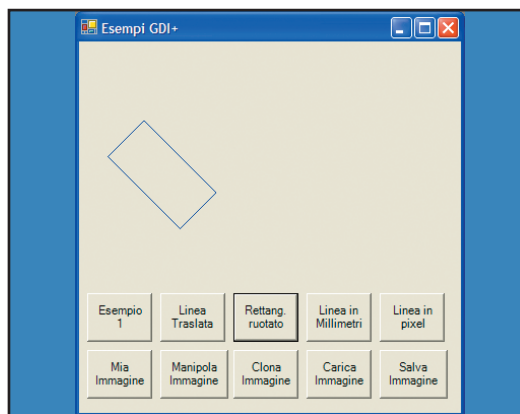
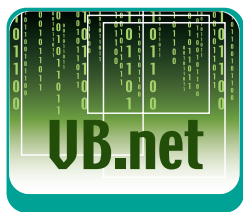



Fig. 3: In figura il rettangolo ruotato

COSA È UN'IMMAGINE BITMAP

Una immagine bitmap è costituita da una matrice di bit che permette di specificare il colore di ogni pixel in una matrice rettangolare di pixel. Il numero di colori che è possibile assegnare ad ogni pixel è determinato dal numero di bit destinati al singolo pixel. Se ad esempio ogni pixel è rappresentato da 4 bit, sarà possibile assegnare ad un dato pixel uno dei 16 colori diversi disponibili (2^4 = 16). Il formato BMP è un formato standard utilizzato da Windows per la memorizzazione di immagini indipendenti da periferiche e da applicazioni. Il numero di bit per pixel (1, 4, 8, 15, 24, 32 o 64) per un dato file BMP viene specificato nell'intestazione del file. I file BMP più comuni utilizzano 24 bit per pixel.

MODIFICARE LA SCALA

Sino a quando si utilizza il pixel come unità di misura, le coordinate della periferica corrispondono alle coordinate della pagina. Per applicare una trasformazione di pagina, la classe *Graphics* mette a disposizione le proprietà *PageUnit* e *PageScale*. La proprietà *PageUnit* accetta un valore enumerato *GraphicsUnit* utilizzabile per indicare un'unità di misura diversa dal Pixel (valore di default per il video). Sono ammessi i valori:

- Inch, indica come unità di misura il pollice.
- Millimiter, indica come unità di misura il millimetro.
- Point, indica come unità di misura un punto della stampante, 1/72 di pollice.
- Display, indica come unità di misura 1/75 di pollice.
- Document, indica come unità di misura quella del documento, 1/300 di pollice:

GRAFICA VETTORIALE

La grafica vettoriale implica il disegno di primitive, quali linee, curve e poligoni, specificati da un insieme di punti in un sistema di coordinate. Il disegno non viene rappresentato dalla serie di pixel, che lo compone sullo schermo, ma è possibile, ad esempio, disegnare una linea retta specificando soltanto le coordinate dei due punti estremi, oppure disegnare un rettangolo tramite un punto che ne rappresenti la posizione dell'angolo superiore sinistro ed un paio di numeri che ne indicano la larghezza e l'altezza.
Gli oggetti messi a disposizione da GDI+ per la gestione della grafica vettoriale sono racchiusi nel namespace *System.Drawing.Drawing2D*.

Vogliamo, ad esempio, disegnare una linea che colleghi i punti di coordinate (0,0) e (20,50), dove il punto (20,50) si trova 20 millimetri a destra e 50 millimetri in basso rispetto al punto (0, 0), dovremo quindi scrivere:

```
Dim OggettoGrafico As Graphics = Me.CreateGraphics
Dim OggettoPen = New Pen(Color.Red)
OggettoGrafico.PageUnit = GraphicsUnit.Millimeter
OggettoGrafico.DrawLine(OggettoPen, 0, 0, 20, 50)
OggettoGrafico.Dispose()
```

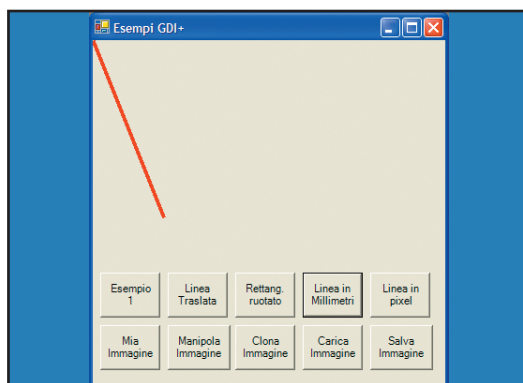



Fig. 4: In figura la retta nella nuova scala

Dopo aver eseguito questo codice, possiamo notare come sia aumentato lo spessore della linea disegnata. Questo effetto si verifica perché gli oggetti *Pen* (per default) hanno una larghezza di una unità, misurata rispetto alle coordinate della periferica. Per questo, adottando i millimetri come unità di misura, la linea è stata disegnata con un oggetto *Pen* largo un millimetro. Per evitare questo effetto, possiamo utilizzare le proprietà di sola lettura *DpiX* e *DpiY* che permettono di ottenere la risoluzione orizzontale e verticale, in modo da determinare il valore da passare al costruttore dell'oggetto *Pen:*

```
Dim OggettoGrafico As Graphics = Me.CreateGraphics
OggettoGrafico.PageUnit = GraphicsUnit.Millimeter
Dim OggettoPen = New Pen(Color.Red, 1 /
                                OggettoGrafico.DpiX)
OggettoGrafico.DrawLine(OggettoPen, 0, 0, 20, 50)
OggettoGrafico.Dispose()
```

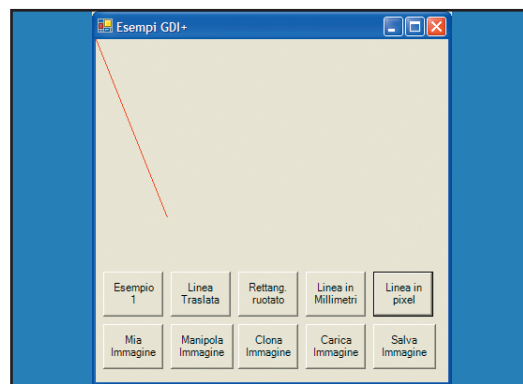



Fig. 5: In figura la retta con lo spessore modificato

Infine, la proprietà *PageScale* permette di ridurre o ingrandire di un dato fattore le unità di misura globali e quelle della pagina.

GESTIONE DELLE IMMAGINI

La tecnologia GDI+ prevede un sottoinsieme creato appositamente per mostrare, ed elaborare, tanto immagini raster (bitmap) che immagini vettoriali (metafile). Gli oggetti più rilevanti per l'elaborazione delle immagini sono:

- **Image** - classe base astratta, che fornisce metodi per salvare e caricare immagini da disco, ed altre operazioni sulle immagini
- **Bitmap** - deriva da *Image*, e consente di espanderne le funzionalità fornendo metodi aggiuntivi per modificare le immagini raster e per accedere ai singoli pixel dell'immagine. La classe *Bitmap* supporta numerosi formati di file, compresi BMP, GIF, JPEG, PNG e TIFF.
- **Metafile -** permette di espandere le funzionalità

della classe *Image* fornendo metodi aggiuntivi per salvare, caricare, modificare ed esaminare immagini di tipo vettoriale (memorizzate sotto forma di comandi e impostazioni di disegno). La classe metafile supporta immagini memorizzate nei formati: *WMF(Windows Metafile), EMF(Enhanced Metafile), EMF+*

Riassumendo: per caricare e visualizzare un'immagine vettoriale, sono necessari un oggetto *Graphics* ed un oggetto *Metafile*. Per visualizzare un'immagine raster, sono necessari un oggetto *Graphics* ed un oggetto *Bitmap*. Le classi *Image* e *Bitmap* appartengono al namespace *System.Drawing,* mentre la classe *Metafile* appartiene al namespace *System.Drawing.Imaging*.

IL METODO DRAWIMAGE

L'oggetto *Graphics* mette a disposizione il metodo *DrawImage*, che permette di visualizzare un'immagine ricevuta da un oggetto *Metafile* o *Bitmap* passato come argomento. Esistono numerose versioni di overload (circa trenta) del metodo *DrawImage,* quindi è possibile fornire argomenti in svariati modi. La versione più semplice accetta come argomenti l'immagine da mostrare a video e le coordinate dell'angolo superiore sinistro dell'area di destinazione. La prima operazione da compiere è quella di caricare un'immagine in un oggetto *Bitmap* (o *Metafile*), per questo possiamo utilizzare il relativo costruttore e scrivere:

```
Dim OggettoBitmap As New Bitmap(
                              "C:\MiaImmagine.jpg")
```

In cui *C:\MiaImmagine.jpg* è il percorso completo del file contenente l'immagine da mostrare. Dopo la creazione dell'oggetto *Bitmap,* è necessario passare questo oggetto come argomento al metodo *DrawImage* insieme alle coordinate. Quindi, per disegnare l'immagine bitmap con l'angolo superiore sinistro in corrispondenza del punto di coordinate (50, 50) possiamo scrivere:

```
Dim OggettoGrafico As Graphics = Me.CreateGraphics
Dim OggettoBitmap As New Bitmap(
                              "C:\MiaImmagine.jpg")
OggettoGrafico.DrawImage(OggettoBitmap, 50, 50)
OggettoBitmap.Dispose()
OggettoGrafico.Dispose()
```

Fig. 6: In figura l'immagine disegnata

Una delle versioni di overload, del metodo *DrawImage*, riceve come argomenti, un oggetto *Bitmap,* contenente l'immagine da disegnare, ed un oggetto *Rectangle,* che permette di specificare il rettangolo in cui verrà tracciata l'immagine. Se le dimensioni del rettangolo di destinazione non corrispondono alle dimensioni dell'immagine originale, l'immagine verrà ridimensionata ed adattata al rettangolo di destinazione. Si può utilizzare questo modo di fare, per ingrandire o comprimere un'immagine.

GRAFICA RASTER (IMMAGINI)

In un'immagine raster, lo spazio è suddiviso in migliaia di quadratini detti pixel. Ogni quadratino può essere un colore, ed è compito del programma di grafica impostare il colore di ogni quadratino in base al tipo di strumento di disegno. Le immagini di questo tipo vengono memorizzate come bitmap, in cui ogni colore dei singoli punti sullo schermo viene trasformato in numero e memorizzato in una matrice. Ogni elemento nella matrice, è accessibile tramite una coppia di coordinate x-y, che identifica il singolo pixel. Gli oggetti messi a disposizione da GDI+ per la gestione delle immagini sono racchiusi nel namespace System.Drawing .Imaging.

ESEMPIO DI CODICE

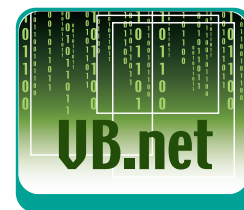
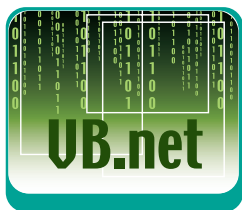
Si dichiara la solita variabile oggetto *OggettoGrafico* di tipo *Graphics,* e la variabile *OggettoBitmap* di tipo *Bitmap* passando come argomento, il percorso completo del file contenente l'immagine da mostrare

```
Dim OggettoGrafico As Graphics = Me.CreateGraphics
Dim OggettoBitmap As New Bitmap(
                              "C:\MiaImmagine.jpg")
```

Possiamo utilizzare la proprietà *Width* dell'oggetto *Bitmap* che specifica la larghezza dell'immagine e la proprietà *Height* che ne specifica l'altezza, per definire un rettangolo delle esatte dimensioni dell'immagine. Disegniamo questo rettangolo nel punto (0,0)

```
Dim RettangoloEsatto As New Rectangle(0, 0,
          OggettoBitmap.Width, OggettoBitmap.Height)
```

Dividendo per due la larghezza e l'altezza dell'immagine, possiamo definire un rettangolo dalle di-

mensioni dimezzate dell'immagine. Pertanto, l'immagine risulterà compressa. Disegniamo questo rettangolo nel punto (150,0)

```
Dim RettangoloCompresso As New Rectangle(150, 0,_
  OggettoBitmap.Width / 2, OggettoBitmap.Height / 2)
```

Moltiplicando per due la larghezza e l'altezza dell'immagine, possiamo definire un rettangolo delle dimensioni uguali al doppio dell'immagine, pertanto l'immagine risulterà ingrandita rispetto alle dimensioni originali. Disegniamo questo rettangolo nel punto (230,0)

```
Dim RettangoloEspanso As New Rectangle(230, 0, _
  OggettoBitmap.Width * 2, OggettoBitmap.Height * 2)
```

Fig. 7: In figura le tre immagini di dimensioni diverse

Si mostrano a video le tre immagini, utilizzando il metodo *DrawImage*

```
OggettoGrafico.DrawImage(OggettoBitmap,
                                    RettangoloEsatto)
OggettoGrafico.DrawImage(OggettoBitmap,
                                 RettangoloCompresso)
OggettoGrafico.DrawImage(OggettoBitmap,
                                   RettangoloEspanso)
```

Si rilasciano i riferimenti agli oggetti

```
    OggettoBitmap.Dispose()
    OggettoGrafico.Dispose()
```

NOTA

Utilizzando il metodo *ScaleTransform*, è possibile modificare la scala utilizzata, lungo i due assi, per disegnare sull'oggetto *Graphics*. I fattori di scala in direzione *x* ed *y* devono essere passati come argomento.

CONTROLLI, METODI E PROPRIETÀ

Il controllo *OpenFileDialog* permette di visualizzare la finestra di dialogo *Apri*, per utilizzarlo si deve selezionare il controllo *OpenFileDialog* e disegnarlo sulla form, se non si agisce sulla proprietà *Name* il nome del controllo sarà *OpenFileDialog1*. Per visualizzare la finestra di dialogo *Apri*, si deve utilizzare il metodo *ShowDialog*. La proprietà *FilterIndex* permette di impostare l'estensione dei file che sarà possibile visualizzare, si possono anche definire dei filtri multipli, in questo caso ogni filtro deve essere separato dalla barra verticale, ad esempio: Files di testo *(*.txt)|*.txt|Tutti i files (*.*)|*. Per stabilire quale file è stato selezionato dall'utente, si deve utilizzare la proprietà *FileName*.**

IL METODO CLONE

La classe *Bitmap*, mette a disposizione il metodo *Clone*, che permette di creare una copia di un oggetto *Bitmap* esistente. Il metodo Clone riceve come parametri:

- Un oggetto *Rectangle*, relativo al rettangolo di origine, che permette di determinare quale porzione della bitmap originale dovrà essere copiata.
- Un valore di tipo *PixelFormat*, che definisce il formato dei dati relativi al colore per ciascun pixel dell'immagine (in particolare definisce il numero di bit di memoria associati ad ogni pixel)

Vogliamo, ad esempio, creare un nuovo oggetto *Bitmap* mediante la clonazione della metà superiore di un oggetto *Bitmap* esistente, il codice sarà il seguente:

```
Dim OggettoGrafico As Graphics = Me.CreateGraphics
Dim OggettoBitmap As New Bitmap(
                                "C:\MiaImmagine.jpg")
Dim RettangoloSorgente As New Rectangle(0, 0,
      OggettoBitmap.Width, OggettoBitmap.Height / 2)
Dim OggettoBitmapClonato As Bitmap =
            OggettoBitmap.Clone(RettangoloSorgente, _
                  Drawing.Imaging.PixelFormat.DontCare)
    OggettoGrafico.DrawImage(
                          OggettoBitmapClonato, 10, 10)
    OggettoBitmap.Dispose()
    OggettoBitmapClonato.Dispose()
    OggettoGrafico.Dispose()
```

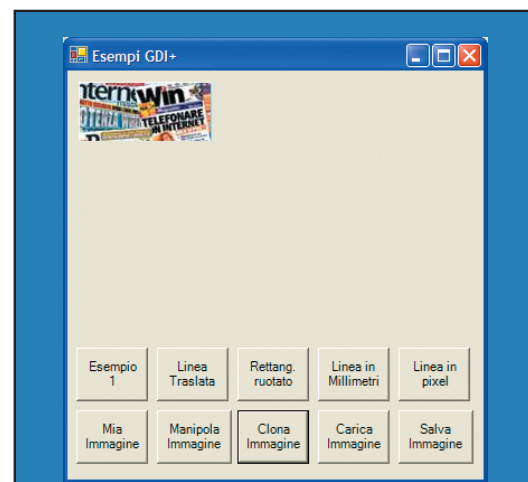



Fig. 8: In figura la porzione di immagine clonata

Caricare e salvare un'immagine da un file qualsiasi
I metodi *FromFile* e *LoadFromStream*, delle classi *Image* e *Bitmap*, consentono di caricare un'immagine da un file o da un oggetto *Stream* aperto, dove per *Stream* si intende una generica sequenza di byte (flusso). Un flusso di byte non deve necessariamente provenire da un file ma ad esempio da una qualsiasi periferica di input/output, o da un socket TCP

/IP. L'oggetto *Stream* permette la visualizzazione generica di questi diversi tipi di flusso, senza che lo sviluppatore venga a contatto con i dettagli specifici del sistema operativo e con le periferiche sottostanti. Possiamo scrivere una procedura, che utilizzi il metodo *FromFile* ed un controllo *OpenFileDialog* per richiedere all'utente il nome del file dell'immagine da visualizzare. Al solito utilizziamo il metodo *DrawImage* per mostrare l'immagine sullo schermo. Il codice da scrivere sarà il seguente

```
Private Sub MostraImmagineDaFile()
    'si crea un oggetto Graphics
    Dim OggettoGrafico As Graphics =
                                Me.CreateGraphics
    'filtra i tipi di file selezionabili dall'utente
    OpenFileDialog1.Filter = "Image files|*.bmp;
                     *.gif;*.jpg;*.jpeg;*.tif;*.png;"
    'controllo sull'avvenuta selezione di un file
    If OpenFileDialog1.ShowDialog =
                           DialogResult.OK Then
        'crea un oggetto Bitmap e ci carica un file
        Dim OggettoBmp As Bitmap =
              Bitmap.FromFile(OpenFileDialog1.FileName)
        'disegna l'immagine contenuta nell'oggetto
                                             Bitmap
        OggettoGrafico.DrawImage(OggettoBmp, 0, 0)
        'distrugge l'oggetto Bitmap
        OggettoBmp.Dispose()
        'distrugge l'oggetto Graphics
        OggettoGrafico.Dispose()
    End If
End Sub
```

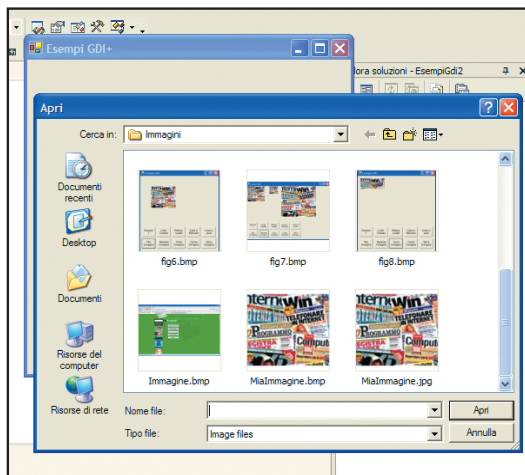
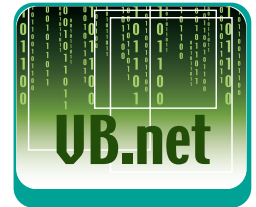



Fig. 9: *In figura l'immagine selezionata*

Il metodo *Save*, delle classi *Image* e *Bitmap*, permette di salvare un'immagine memorizzata nell'oggetto corrispondente. Questo metodo accetta come argomenti: il nome del file ed il formato di destinazione. Utilizzando un controllo *SaveFileDialog* possiamo scrivere una procedura che permetta di salvare un'immagine contenuta in un oggetto *Bitmap*, chiedendo all'utente di specificare il formato desiderato.

```
Private Sub SalvaImmagine()
  Dim OggettoGrafico As Graphics = Me.CreateGraphics
  Dim OggettoBitmap As New
                    Bitmap("C:\MiaImmagine.jpg")
  OggettoGrafico.DrawImage(OggettoBitmap, 0, 0)
  With SaveFileDialog1()
      'titolo della finestra di dialogo
      .Title = "Seleziona il formato del file di
                                  destinazione"
      'filtri delle possibili estensioni
      .Filter = "Bitmap|*.bmp|GIF|*.gif|JPEG|*.jpg"
      'chiede all'utente se sovrascrivere il file,
                                  qualora l'utente
      'abbia specificato un nome file esistente;
      .OverwritePrompt = True
      If .ShowDialog = DialogResult.OK Then
          'controllo sull'estensione selezionata
          Select Case System.IO.Path.GetExtension(
                              .FileName).ToUpper
            Case ".BMP"
            OggettoBitmap.Save(.FileName, System.
                 Drawing.Imaging.ImageFormat.Bmp)
            Case ".GIF"
            OggettoBitmap.Save(.FileName, System
                 .Drawing.Imaging.ImageFormat.Gif)
            Case ".JPG"
            OggettoBitmap.Save(.FileName, System
                .Drawing.Imaging.ImageFormat.Jpeg)
            Case Else
                MessageBox.Show("Formato non
                      ammesso", "Attenzione",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error)
          End Select
      End If
  End With
  OggettoBitmap.Dispose()
  OggettoGrafico.Dispose()
End Sub
```

Luigi Buono

Fig. 10: *L'applicazione in esecuzione*

NOTA

Un'ulteriore versione del metodo DrawImage permette di specificare, oltre al rettangolo di destinazione, anche il rettangolo di origine. Il parametro relativo al rettangolo di origine permette di determinare quale porzione dell'immagine originale dovrà essere disegnata. Utilizzando le proprietà Width ed Height si può ritagliare una parte dell'immagine originale. Anche in questo caso, se le dimensioni del rettangolo di destinazione non corrispondono alle dimensioni del rettangolo di origine, l'immagine verrà ridimensionata e adattata al rettangolo di destinazione.

VmWare 5.0 Un solo PC tanti sistemi

Un emulatore incredibilmente efficace che consente di suddividere le risorse di una macchina fisica su molte macchine virtuali, dando l'illusione si avere a disposizione un sistema completo

Si tratta di una normale applicazione che gira in una finestra del sistema operativo Host. Il sistema su cui l'applicazione viene lanciata, può essere ad esempio Windows, ma ache Linux. La cosa interessante è che l'applicazione VmWare crea una vera e propria macchina virtuale all'interno della quale è possibile installare un secondo sistema operativo. La macchina virtuale sfrutterà parte della memoria della macchina fisica, condividerà con essa la scheda di rete e simulerà gli Hard Disk associandoli a file sulla macchina fisica.

PERCHÉ USARLO?

Spesso c'è bisogno di testare il comportamento di un software rispetto ai diversi sistemi operativi e non c'è possibilità di avere hardware a sufficienza per installare tutti i sistemi, così come è difficile ottenere le stesse configurazioni hardware per ogni macchina. Con VmWare invece è sufficiente creare la macchina virtuale e poi clonarla installando su di essa il sistema operativo su cui si intende effettuare il test, essendo inoltre certi che la configurazione hardware del sistema è analoga su tutte le macchine virtuali. La clonazione di una macchina è un'operazione banale, è sufficiente duplicare i file che la compongono.

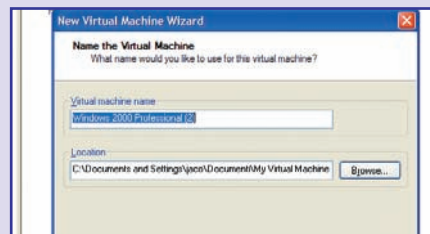
VANTAGGI DI VMWARE

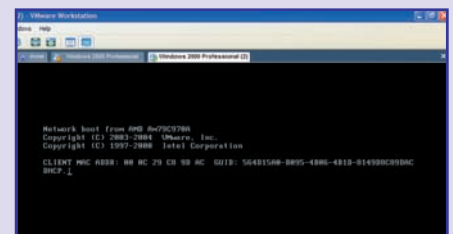
Uno dei vantaggi più interessanti di VmWare è quello di poter sospendere il sistema in un determinato stato. La macchina virtuale serializzerà l'intero processo in un file, tale che è possibile in un secondo momento ripristinare il suo stato al momento in cui è stato effettuato lo snapshot. Altro vantaggio interessante di VmWare è dato dalla possibilità di "Filmare" quanto sta accadendo nella macchina virtuale per poi "visualizzare" il film in un secondo momento. Si tratta di una caratteristica utile quanto importante per la realizzazione di videoguide. VmWare è incredibilmente stabile. Lo abbiamo usato per testare varie installazioni. Da una macchina linux ne abbiamo create diverse virtuali con Windows XP e Windows 2000. Unica accortezza in questo caso è quella di ricompilare i moduli del kernel necessari. I sistemi si sono rivelati sempre stabilissimi e l'emulazione efficiente.

UNA MACCHINA VIRTUALE IN TRE PASSI

> QUALE SISTEMA

1 Dalla maschera è possibile scegliere quale sistema intendiamo installare nella nuova macchina

> DOVE SARANNO I FILE?

2 Scegliamo una locazione per salvare i file che costituiranno il sistema virtuale

> EFFETTUIAMO IL BOOT

3 All'avvio della macchina virtuale il PC si presenta come in figura. Come se fosse l'avvio di una macchina fisica

SOFTWARE SUL CD

KERIO MAILSERVER 5.7.1

Completo e affidabile

Davvero ben fatto questo Mail Server prodotto da Kerio. È proprio la completezza il punto di forza di questo prodotto. Prima di tutto si deve considerare l'altro numero di protocolli supportati. SMTP, POP3, Secure POP3, IMAP, Secure IMAP, Webmail /WAPmail, SecureWebmail/WAPmail. Una rapida occhiata nel comodo pannello di amministrazione ci informa anche che il prodotto è fortemente integrato con SpamAssasin e con il motore Antivirus di Mcaffee.

Le altre caratteristiche interessanti sono costituite da un sistema di backup che consente di effettuare una copia del sistema di posta ad intervalli predefiniti, e dallo scheduler che consente di gestire la coda degli invii in modo ottimale. Una nota particolare merita il plugin per la stampa delle mail su un fax remoto. Infine la webmail integrata risulta molto comoda, anche se in questo caso c'è da segnalare la pecca dell'assenza di un editor Html.

Directory /Kerio

ACQUA DATA STUDIO 4.5.2

Il SQL Manager è servito

Acqua Data Studio è uno strumento decisamente evoluto. Scritto in Java si configura come un SQL Manager completo e affidabile. Supporta un gran numero di DBServer, da Oracle a MsSQL a MySQL.

Consente la gestione totalmente grafica delle tabelle, dei database, delle interrogazioni SQL. Nel nostro caso lo abbiamo usato in alternativa a PHP-MyAdmin, trovandolo utile come sistema da utilizzare in modo standalone, ancora leggermente inferiore per quanto riguarda il numero di funzionalità esposte. D'altra parte PHPMyAdmin è un fuoriclasse nel suo genere ed è tarato su MySQL, mentre AcquaData Studio è del tutto generico. In ogni caso si tratta di un sistema che vi può trarre fuori dai guai in più di una situazione. Da provare!

Directory /AcquaDataStudio

APACHE 2.0.55

Il Web Server Essenziale

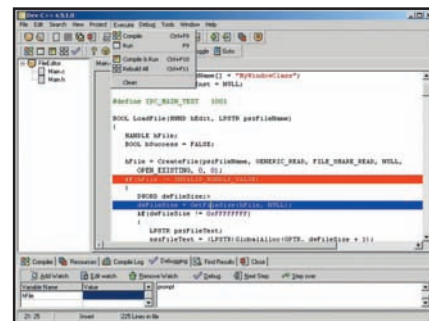
Se siete dei programmatori Web, molto probabilmente avrete bisogno di Apache. Si tratta di uno dei web Server più utilizzati al mondo. Da installare per provare in locale le vostre applicazioni PHP o il comportamento dei vostri siti dinamici.

Non potete usarlo se siete dei programmatori .NET, in tal caso dovrete installare IIS, ma questo è possibile solo con Windows XP pro o superiore e non con la versione Home, perciò in più di una occasione Apache rappresenta un'ottima via d'uscita per i vostri test sul web. Senza contare che è senza dubbio il Web Server più diffuso al mondo e il comprendere le caratteristiche interne del prodotto certamente è un vantaggio non da poco, visto che per numero di funzionalità e qualità non è sicuramente secondo a nessuno.

Directory /Apache

DEV C++ 4.9.9.2

L'IDE per i programmatori C++

Dev C++ è un IDE completo, dotato di tutte le più moderne funzionalità degli ambienti più blasonati. È compatibile con Visual Studio, ma trova la sua maggiore applicazione in congiunzione al compilatore MingGW. È uno degli ambienti più amati dai programmatori C++, per la sua leggerezza e completezza. Si tratta inoltre di un prodotto OpenSource il cui codice scritto in Delphi è liberamente disponibile.

Rappresenta dunque un'ottima opportunità sia per coloro che necessitano di un IDE evoluto per la programmazione giornaliera, ma anche per coloro che vogliono capire come vengono strutturati progetti del genere.

Directory /devcpp

ECLIPSE 3.1.1

Il top per i programmatori Java e non solo

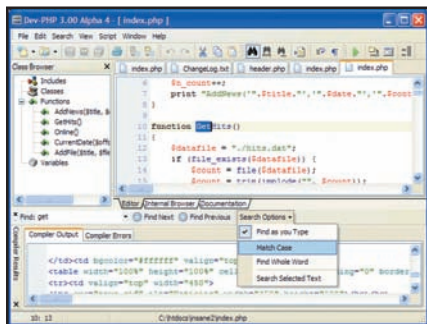
Su Eclipse si sta creando una community interessantissima di aziende e utenti che contribuiscono al suo sviluppo. Se da un lato la sua affidabilità come IDE per i programmatori Java è ormai comprovata, è anche vero che di giorno in giorno l'ambiente viene esteso con nuove funzionalità.

È molto recente la notizia che Macromedia sta sviluppando un plugin basato su Eclipse che consentirà anche ai programmatori Linux e a coloro che non dispongono dell'ambiente completo di programmare siti e applicazioni con tecnologia Flash, utilizzando Eclipse. Si tratta dunque di un prodotto da tenere in grande considerazione sia per i programmatori Java che per gli altri, anche se necessita di hardware consistente, di fatto il suo più grosso Handicap consiste in una certa pesantezza dell'ambiente

Directory /eclipse

DEVPHP 2.0.12

Ti aiuta nella programmazione giornaliera

PHP è un linguaggio completo, maturo, che da solo tiene in piedi gran parte di internet come la conosciamo oggi. Uno dei maggiori Handicap del linguaggio sta nella scarsa presenza di ambienti ti programmazione affidabili e a basso costo.

Questo problema viene risolto da DevPHP che è un IDE completo di syntax highlighting e code complexion. Se pur con qualche difficoltà in più di configurazione consente anche il debugging delle applicazioni e un'elevata integrazione con un completo ambiente di test

Directory /devphp

MYSQL JAVA CONNECTOR

Il provider per connettere Java a Mysql

I database sono sempre in qualche modo coinvolti nello sviluppo. Il Java Connector contiene tutte le funzioni di riferimento per utilizzare MySQL da applicazioni Java. In questo stesso numero di ioProgrammo in uno dei tanti tutorial che caratterizzano questa edizione trovate la guida passo passo a come utilizzarlo

Directory

J2SE 1.5.0.5

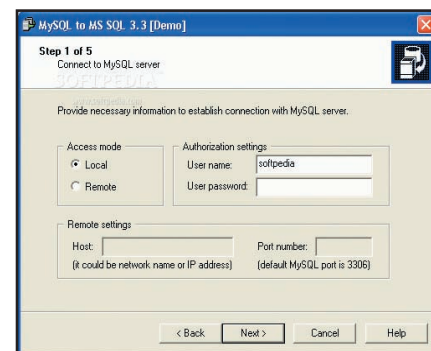
Il framework per Java

Indispensabile per i programmatori Java, il J2SE contiene tutte le classi e gli strumenti per poter realizzare un'applicazione con questo linguaggio. Il J2SE rappresenta la base da cui partire se si vuole sviluppare un qualunque prodotto sia di dimensioni ridotte che di grandi dimensioni. Le varie versioni rilasciate in sequenza segnano anche l'evoluzione di Java come piattaforma di programmazione, che ormai detiene quote di mercato talmente elevate da rappresentare un Must per chiunque in un qualche modo si affacci al mercato della programmazione

Directory /J2SE

MYSQL MIGRATION TOOLKIT

Per migrare verso MySQL

Un comodo Wizard che in pochi passi vi consentirà di spostare i vostri dati da un qualunque database al nuovo MySQL.

MySQL 5.0

Finalmente la nuova versione!

Gli sviluppatori da sempre hanno fatto una colpa a MySQL per l'assenza di strumenti quali stored procedure e trigger. Pur essendo il database più usato in ambiente web, pur essendo estremamente semplice e veloce, riferendosi a MySQL si è sempre detto che l'assenza di questi strumenti non lo configurava come un database professionale. Da questa versione in poi non si potrà più dire niente del genere. MySQL 5.0 rappresenta un salto di continuità rispetto al passato. Sono stati aggiunte proprio le funzioni relative a Stored Procedure e ai Trigger, che vanno a completare la nutrita schiera di funzionalità che già caratterizzava il prodotto. Se non avete già pensato a una migrazione dalle vecchie versioni forse è il caso di farlo!

Directory: /mysql

Si tratta di un tool molto interessante che rappresenta la volontà del team di MySQL di proporsi come una controparte affidabile, senza soffrire di complessi di inferiorità rispetto a colossi del settore quali MsSQL, Oracle e gli altri.

Directory /MySQL

MYSQL DOT NET CONNECTOR

Il provider per connettere .NET a MySQL

Non ha ancora tutte le caratteristiche evolute dei provider tipici integrati in Visual Studio, manca ancora una certa integrazione e si vede ad esempio nel mancato uso dei Wizard, ma questo provider per dot net, rappresenta una delle poche soluzioni gratuite per utilizzare db MySQL da applicazioni .NET. Se si pensa che Microsoft è anche proprietaria di uno dei database direttamente concorrenti, non è cosa da poco

Directory /MySQL Net Connector

NHIBERNATE 1.0.10

La persistenza dei dati versione .NET

Hibernate è uno dei tool di persistenza più noti al mondo. Consente di mappare oggetti a dati tipicamente strutturali come quelli esposti da SQL, e infine di mantenerli persistenti. NHIbernate è il porting di questo software per la piattaforma .NET. Un'applicazione che certamente, nonostante, una certa complessità, risolve alcuni dei problemi che maggiormente affliggono i programmatori in fase di stesura di applicazioni che si connettano a DB

Directory /NHIbernate

PHP 4.4.1/5.0.5

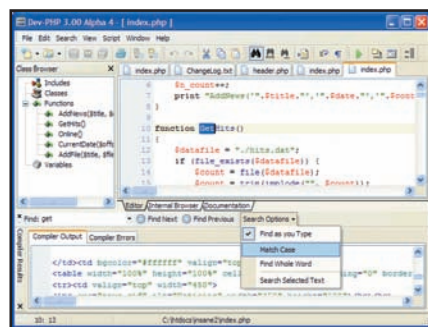
Il linguaggio del Web

Inutile dirlo PHP è sicuramente il dominatore incontrastato della rete. La sua natura OpenSource, la sua curva di apprendimento molto bassa, la grande disponibilità di applicazioni e infine la sua alta interazione con MySQL ne fanno un linguaggio estremamente appetibile per il popolo degli internauti, che in effetti l'hanno eletto loro leader.

Directory /PHP

NOTEPADPP

Il più piccolo degli editor

Incredibilmente leggere, notepadPP è identico al notepad di windows, eccetto che per il fatto che è dotato si Syntax HighLighting per quasi tutti i linguaggi e gestisce file di testo anche molto lunghi, proprio perché si configura come un editor specifico per programmatori.

Si tratta di un software da utilizzare quando si devono fare modifiche rapidissime e non si desidera caricare un intero ambiente completo.

Directory /NotepadPP

QCODO

Il PHP Development Framework

Qcodo è un Framework che si pone ad un livello superiore rispetto al linguaggio. Consente di partire dall'analisi del flusso dei dati di un'applicazione per poi evolvere in classi, oggetti e metodi. Si tratta di uno strumento inconsueto per PHP che notoriamente non dispone di strumenti di questa complessità. Sicuramente può essere uno strumento utile per l'analisi funzionale e la buona progettazione di un'applicazione

Directory: /Qcodo

ULTIMATE ++

Il nuovo IDE per C++

Completo e ben fatto questo IDE che si affaccia alla programmazione tentando di rubare la scena al ben più noto DEV C++. Dotato di code complexion e sintax highliting, un ottimo debugger, è votato persino ad essere un ambiente RAD programmabile a mezzo form come accade negli ide più blasonati. Certo in questo senso ha ancora molto da crescere, tuttavia si pone come un ottimo IDE freeware e alternativo a Dev C++

Directory: /Ultimate

ZNF 0.7.6

Per implementare il pattern MVC in PHP

Molti di voi conosceranno STRUTS, uno dei framework più noti in ambiente Java per la costruzione di applicazioni Web legate al pattern MVC, Model View Controller. A fronte di una leggera difficoltà iniziale, il pattern MVC garantisce una completa separazione dell'aspetto grafico di un'applicazione, dal suo motore di controllo, e infine degli oggetti che implementano le funzioni richieste. Una struttura del genere, fa si che applicazioni di grandi dimensioni siano eccezionalmente manutenibili e garantisce uno sviluppo omogeneo e leggibile, per cui sviluppare in questo senso significa seguire una miglior logica strutturale nei propri progetti. ZNF aiuta a creare applicazioni che seguono questo pattern e si pone come framework avanzato per la costruzione di applicazioni PHP di classe enterprise

Directory: /Znf

TURBO ADMIN

La soluzione AJAX per i database

AJAX è una tecnologia che si sta diffondendo rapidamente per la sua capacità di aggiornare solo le parti essenziali di una pagina WEB piuttosto che aggiornarla integralmente. Questo aumenta notevolmente la velocità della pagina inoltre consente di realizzare applicazioni più facilmente fruibili dagli utenti. Turbo Admin è una mini applicazione che consente di esplorare un database utilizzando la tecnologia AJAX con PHP.

Directory:

SDL.NET

La libreria di videogiochi per .NET

Si tratta di un insieme di componenti che effettuano un binding fra .NET e la ben nota libreria SDL. Grazie a questi componenti altamente specializzati è possibile un accesso rapido ed efficace a parti hardware come per esempio scheda audio, scheda video, mouse joystick. Tutti elementi che possono essere pilotati per realizzare giochi in modo efficace

Directory: /Sdlnet

Fattorizzazione di numeri

Trovare il prodotto di fattori di un numero intero è un compito niente affatto banale, soprattutto è anche la base dei metodi di crittografia a chiave pubblica.

REQUISITI

Conoscenze richieste
Basi di programmazione C++

Software
-

Impegno

Tempo di realizzazione

Codici, alfabeti, linguaggi e molti degli elementi che compongono il background del programmatore hanno come essenza i numeri. Nello scorso appuntamento abbiamo potuto osservare come i numeri primi, la cui memoria risaliva per molti di noi all'educazione elementare, siano oggetti di fondamentale importanza. Infatti, il metodo di crittografia a chiave pubblica RSA si basa sulla manipolazione di due numeri primi. Ma un'altra componente è alla base dello stesso algoritmo: si tratta proprio della scomposizione in fattori del numero di grandi dimensioni. È seguendo questa linea immaginaria che si poggia sull'approfondimento del metodo RSA, che siamo giunti a studiare le implicazioni che derivano dalla manipolazione a volte elementare di numeri. In questo ambito vedremo che ruolo gioca il computer. Il problema odierno è quello di fattorizzare un numero, ovvero di trovare due numeri il cui prodotto dia il numero stesso. Un primo ostacolo alla ricerca è dato dall'eventualità che il numero sia primo, in tal caso non è possibile risolvere il problema a meno di non considerare la soluzione degenere *1*n*, che ovviamente non teniamo nemmeno in considerazione. Altri metodi per così dire "rudimentali" fanno dei tentativi con la divisione del numero per i primi numeri primi (scusate il gioco di parole). Tale pratica, statisticamente, dà risultati in molti casi, purtroppo si tratta però di soluzioni che potremmo definire di serie B. Infatti, non hanno un grande interesse, soprattutto nel campo informatico, fattorizzazioni per le quali uno dei due numeri è molto più piccolo del secondo. Così, nel corso degli anni, si sono succeduti molti studi sull'argomento con pregevoli risultati. Insieme ne esamineremo qualcuno con particolare attenzione al risvolto algoritmico.

SOLUZIONE PER DIFFERENZE QUADRATICHE

Il primo metodo, dovuto a Fermat, è il frutto di una sfida. Un altro importante matematico del XVII secolo, Mersenne, in una missiva provocava Fermat chiedendogli di fattorizzare il numero

FATTORIZZAZIONE ED RSA

RSA funziona come segue:

1. **Scelgo due grandi numeri primi *p* e *q* (ad esempio 1024 bit ciascuno) e calcolo i prodotti *n=pq* e *m= (p-1)(q-1)*.**
2. **Scelgo un numero d minore di *m* ma che rispetto ad esso sia un primo relativo. Significa che *m* e *d* non hanno fattori primi in comune. Entrambi i numeri non sono primi e *m* è sicuramente pari, si tratta infatti del prodotto di due numeri pari.**
3. **Scelgo ancora un altro numero e per assicurare la divisibilità di *(de – 1)* per *m*, che matematicamente scriviamo come *de = 1 (mod m)*.**
4. **Rendo pubbliche le due chiavi *n* ed *e*. Altri membri della rete possano tenerle senza peraltro mantenerle segrete.**
5. **Custodisco segretamente le chiavi: *p, q, m* e *d*.**

Modalità di invio di un messaggio in forma segreta. Si trasmette un numero *T* minore di *n*, che con opportune manipolazioni può corrispondere a un testo o a parte di esso trasformandolo come segue:

$C=T^e \mod n$

Il risultato *C* è il codice segreto *(chiper text)* che ci verrà inviato. La funzione di decodifica o decrittografia è fatta utilizzando le chiavi private.

$T=C^d \mod n$

Si ottiene quindi il testo in chiaro *(plain text)* originariamente trasmesso. La scomposizione in fattori è molto complicata per numeri molto grandi, anche la potenza di computazione di super elaboratori non aiuta. E comunque anche i numeri d e m rimangono privati.

100.895.598.169. Per l'epoca tali circostanze erano comuni, ed è grazie ad esse che la matematica ha visto una sostanziale evoluzione. Oggi lo studio della matematica è mosso da nuove sfide. L'idea alla base della soluzione del matematico francese era di tentare di scrivere il numero come differenza di quadrati.

$N=x^2-y^2 = (x+y)(x-y)$

Ovviamente, tale forma garantisce anche la scrittura del numero come prodotto di due numeri, i binomi *(x+y)* e *(x-y)*, che conducono in definitiva al nostro obiettivo. Il procedimento proposto è per tentativi. Vedremo che tutti i metodi sono per tentativi, che per la situazione odierna, nella quale si può contare su potenti calcolatori, non dovrebbero spaventare. Si prova con un primo valore di *x*, poi vedremo i metodi di scelta, e si verifica se lo scarto del numero *N* rispetto al quadrato di *x* è a sua volta un quadrato. Si prova, in altri termini la relazione scritta sopra. Se così non fosse si seleziona un nuovo valore di *x* e si continua. Il processo si ferma quando lo scarto genera un quadrato, abbiamo cioè trovato la *y*, quindi la soluzione. Entriamo adesso nel merito dell'algoritmo. Il primo passo è scrivere il nostro numero *N* di cui si vogliono calcolare due fattori nella forma:

$N=x^2-z$

Il metodo come vedremo valuterà diversi valori di *x* fin quando *z* non diventa un quadrato e quindi rientra nella forma descritta ad inizio paragrafo. Il primo valore di *x* che si tenta è:

x=[sqrt(N)] + 1

Con *sqrt* radice quadrata e con la parentesi quadra che indica la parte intera. A tale proposito è necessaria una piccola puntualizzazione: si deve ricordare che stiamo trattando soltanto con numeri interi, senza la virgola, e che le nostre elaborazioni non prevedono mai la presenza di numeri reali, con virgola. Per comodità indichiamo tale radice con il simbolo *n* (questa volta minuscolo).

n=[sqrt(N)]
$N=n^2+r$

Quindi il resto o scarto rispetto al quadrato è *r*. Facciamo i calcoli per verificare il più piccolo valore di *x* e *z* che può essere testato.

$z=x^2-N = (n+1)^2-n^2-r = 2n+1-r$
$x=n+1$

Il passo successivo è testare la soluzione *z*. Se si tratta di un quadrato abbiamo terminato, altrimenti si propongono nuovi valori di *x* (Fermat semplicemente proponeva di incrementare di uno) e verificare i conseguenti valori di *z*. Applichiamo un esempio per capire meglio il procedimento. Analizziamo lo stesso numero che propose Fermat per spiegare il metodo. Si considera *N=2027651281* al primo passo si ottiene *n= 45029*, *r=40440*, *z=49619* e *x=45030*. Si può facilmente verificare che *z* non è un quadrato. Lo faremo con il programmino proposto di seguito, ma anche una semplice calcolatrice può bastare. Certo è che ai tempi di Fermat tante "comodità" non esistevano cosicché si usavano altri metodi. Si itera il procedimento e si prova una seconda soluzione. Si incrementa la *x* e si ottiene conseguentemente un nuovo valore numerico di *z*.

$z=(x+1)^2-z=(x+1)^2-x^2+z=2x+1+z$
*z=2*45030+1+49619=139680*
x=45031

Ahimé, il numero *z* ancora non è quadrato, basta osservare che le ultime due cifre (80), esse non sono un quadrato. Procedendo così ad un certo punto (vedremo quale nel prossimo paragrafo) si ottiene un valore di *z* che è finalmente un quadrato, si tratta di *z=1040400* che è quadrato di 1020. Riprendendo la relazione: *N=x2-z*, possiamo finalmente esprimere *N* come la differenza tra due quadrati le due radici corrispondenti sono *x=45041* e radice di *z* che è appunto 1020, che chiameremo *y*. Così i due fattori prodotti sono *(x+y)* e *(x-y)*, ovvero *45041+1020 = 46061* e *45041-1020=44021*. In definitiva si può verificare che il risultato è:

*N=(x+y)(x-y)=46061*44021= 2027651281*

PICCOLI TRUCCHI USATI NELL'ERA DELLA PENNA

Il metodo usato da Fermat per verificare se un numero è un quadrato, si basa sull'analisi delle ultime due cifre del numero. Se queste sono un quadrato, si tratta di un quadrato, no altrimenti. Molti studiosi odierni sono comunque curiosi di conoscere tutte le regole o trucchi che venivano usati dal matematico francese. Riusciva a procedere con una significativa velocità i più complessi calcoli numerici che si trovava ad affrontare.

LA SOLUZIONE COME PROGRAMMA

Chissà cosa mai avrebbe potuto produrre Fermat se avesse avuto a disposizione un calcolatore. Ad ogni modo la codifica del suo metodo risulta alquanto agevole. Si tratta di predisporre un ciclo

con il quale valutare sempre nuovi valori di x e riportare i risultati di interesse in output. Ecco la routine che attua il metodo. Per semplicità sono state usate variabili con gli stessi nomi dello studio algebrico.

```
const int nmax=15;
     long N,n,z,r,x;
      int nprove;
void Fermat(long NN)
{   cout<<"\n\t"<<" z "<<'\t'<<" x "<<'\t'<<'\n';
    n=sqrt(NN);
    r=NN-n*n;
    x=sqrt(NN)+1;
    z=2*n+1-r;
    for (nprove=0;nprove<=nmax; nprove++)
    {    z=z+2*x+1;
         x=x+1;
         cout<<'\t'<<z<<'\t'<<x<<'\n';
    };
    getch();
};
int main()
{
    cout<<"\tN --> ";
    cin>>N;
    Fermat(N);
    Fermat_e_Lucas(N);
}
```

L'output della funzione proposta per il numero dell'esempio è riportata in **Figura 1**.
Se si vuole uscire dal ciclo quando è stata trovata la soluzione, basta aggiungere un controllo. Inoltre, è necessario modificare la seconda condizione che ci garantisce un numero finito di cicli. A tale proposito sono stati negli anni condotti diversi studi. Il matematico francese come nelle sue abitudini ha svolto un ottimo lavoro producendo un metodo, ma non lo ha completato. Sapere fin quando cercare per essere sicuri che si tratta di un numero primo è fondamentale, a tale scopo ha dato il suo contributo Lucas che partendo dal fatto che un numero primo deve necessariamente rispettare espressioni del tipo:

$X+y=N, x-y=1$ così $x=(N+1)/2$

Ma se $N=ab$ con $a<b$, allora:

$x+y=a, x-y=b$ e $x=(a+b)/2=(a+N/a)/2$

Questo ultimo numero è sicuramente più piccolo di $(N+1)/2$ che a sua volta diventa un ottimo lower bound. Ossia, un limite oltre il quale è inutile controllare. Alla luce di queste nuove considerazioni, costruiamo una nuova funzione C++.

```
bool quadrato(long v)
{   long rv;
    rv=sqrt(v);
    return (rv*rv)==v;
};
void Fermat_e_Lucas(long NN)
{   cout<<"\n\t"<<" z "<<'\t'<<" x "<<'\t'<<'\n';
    n=sqrt(NN);
    r=NN-n*n;
    x=sqrt(NN)+1;
    z=2*n+1-r;
    for (nprove=0;(!quadrato(z)) && (x<(N+1)/2);
                                        nprove++)
    {    z=z+2*x+1;
         x=x+1;
         cout<<'\t'<<z<<'\t'<<x<<'\n';
    };
    getch();
};
```

La funzione quadrato restituisce un valore booleano che sarà vero se il numero esaminato è un quadrato. Per verificarlo si estrae prima la radice intera e poi si verifica se il quadrato è uguale al numero iniziale. Per questi compiti il linguaggio C++ risulta molto efficiente. In assegnazione del tipo *rv=sqrt(v)* si azionano le regole di casting. Attraverso conversioni implicite il valore a destra *(r-value)* viene automaticamente convertito nel tipo della variabile a sinistra *(l-value)*. Nel caso specifico, la conversione avviene mediante un troncamento della parte decimale, che è ciò che ci serve.

FATTORIZZAZIONE CON RESIDUI QUADRATICI

Si devono a Gauss due metodi di fattorizzazione. Il primo è quello che esamineremo. Il metodo si basa sulla ricerca del più piccolo residuo quadratico del numero che bisogna fattorizzare.
Il numero a è residuo quadratico del numero N, se esiste un numero x, tale che:

$x^2=a \pmod N$

Ricordo che la modalità di scrittura del modulo proprio di lessico matematico equivale alla funzione mod dei tipici linguaggi di programmazione (% per il C++). Così l'espressione (riferita al presente caso) diventa:

$(x^2-a) \bmod N=0$

Il metodo è di esclusione. L'idea si fonda sulla considerazione che se a è un residuo quadratico

NOTA

ALTRI METODI
Il metodo delle differenze quadratiche richiede un gran numero di calcoli. Successori di Fermat hanno tentato altri metodi per scrivere N. Kausler e Collins separatamente hanno tentato delle congruenze modulo 4. N dovrà essere congruente a 1 o a 3 modulo 4. Tra i principali matematici che hanno affrontato il problema ricordiamo Kraitich e Shermar Lermar.

```
\edmaster\ioprogrammo\n_98\sol
 N --> 2027651281

  z       x
 139680  45031
 229743  45032
 319808  45033
 409875  45034
 499944  45035
 590015  45036
 680088  45037
 770163  45038
 860240  45039
 950319  45040
 1040400 45041
 1130483 45042
 1220568 45043
 1310655 45044
 1400744 45045
 1490835 45046
```

Fig. 1: ***Output del programma sviluppato per l'implementazione metodo di fattorizzazione***

di *N*, allora possiamo scartare una parte dei numeri primi come fattori di *N*. Attraverso iterazioni successive è possibile scartare altri insiemi di numeri primi. Alla fine del processo ci si può trovare in due situazioni. La prima è che siano rimasti dei numeri primi, in tal caso è possibile fattorizzare proprio per quei numeri. La seconda è che non sia rimasto alcun numero, si tratta allora di numero primo. Questo metodo presuppone che siamo in grado di escludere i numeri primi. Gauss ha risolto la questione costruendo una tabella per piccoli residui quadratici.
In **Tabella 1** la prima colonna contiene i numeri primi. Nelle successive colonne il simbolo **o** indica che il numero positivo o negativo, intestazione della colonna è un residuo quadratico del numero primo corrispondente alla riga. Quindi *-6, 13, -14, 17, 37* e *-53* sono tutti residui quadratici di 127. Si può rilevare come il metodo abbia un'applicazione molto semplice. Ma, come lo stesso Gauss ha osservato, bisognerebbe trovare un modo per conoscere piccoli residui quadratici. Valgono a tale proposito utili proprietà. Se un numero è residuo quadratico per un numero *Z* lo sarà anche per i suoi multipli. Inoltre, il rapporto e il prodotto di due residui quadratici di *Z* sono anche essi residui quadratici. Si possono quindi combinare residui quadratici di *Z* e dei suoi multipli. Ad esempio, *N=997331*, abbiamo *2N=(1412)2 +918* e *918=2*33*17*, così 6 e 17 sono residui quadratici di *N*. Ma un modo più elegante è trovare il residuo quadrato usando la continua espansione della radice di *N*. Supponiamo di voler analizzare il numero 997331, così come fece Gauss per mostrare il suo metodo. Secondo una regola generale bisognerà verificare i numeri primi iniziali, è sufficiente fermarsi alla radice quadrata di *N*.
Nell'esempio il più grande primo sotto *sqrt(997331)* è 997. Successivamente, componendo i risultati secondo le regole prima descritte di possono eliminare molti numeri primi e raggiungere l'obiettivo.

FATTORIZZAZIONE CON CONTINUE FRAZIONI

Questa idea per completare il metodo di Gauss è dovuta ad un altro grande matematico Legendre. In sintesi, il metodo prevede la continua espansione della radice così come mostrato dalla formula riportata di seguito.

$$\sqrt{N} = b_0 + \cfrac{1}{b_1 + \cfrac{1}{b_2 + \cfrac{1}{b_3 + \cfrac{1}{\dots}}}}$$

La successiva scomposizione del numero iniziale, la radice di N porta alla risoluzione di problemi (calcoli numerici) via via più semplici e quindi ci permette di trovare il risultato. Il problema può essere formalizzato da una serie di formule che indicano i valori generici (iesimi) contenuti dalle variabili in gioco. Essendo valori iesimi si dovrà nell'algoritmo impostare un'iterazione. La formula ricorrente che esprime il processo è:

x0=sqrt(N), bi=[xi]   e xi+1=1/(xi-bi)

La x può essere scritta come xi=(sqrt(N)+Pi)/Qi. Si evidenzia l'importante vantaggio di manipolare i due numeri P e Q che sono degli interi. Sono queste le relazioni alla base del metodo che iterate forniscono i valori della soluzione.

	-6	+13	-14	+17	+37	-53
3	o	o	o		o	o
5	o		o			
7	o		o		o	
11		o	o	o		o
13		o	o	o		o
17		o		o		o
19			o	o		o
23		o	o			o
....						
113		o	o			
127	o	o	o	o	o	o
131	o	o	o			
....						

Tabella 1: Una tabella per i piccoli residui quadratici

CONCLUSIONI

Abbiamo visto come un numero può essere fattorizzato. Le teorie analizzate risalgono a qualche secolo fa, ma sono ancora molto attuali se si pensa che costituiscono lo zoccolo teorico per molti degli studi odierni. Come spesso accade in questo campo, la presenza dell'elaboratore ha dato nuovo vigore alla teoria. Inoltre, la presenza di numerosi algoritmi in cui la fattorizzazione è indispensabile ha spinto verso una nuove soluzioni. Sta di fatto che non esistono algoritmi di estrema efficienza visto che il metodo RSA è ancora sicuro. A tale proposito è utile ricordare che i numeri sono costituiti da un numero sempre maggiore di byte, il che garantisce circa le possibili fattorizzazioni che si possono effettuare in breve tempo da computer sempre più potenti per craccare sistemi di crittografia. Insomma, anche questa volta i "semplici" numeri ci hanno riservato gradevoli sorprese e ci aspettano per un nuovo appuntamento di soluzioni dove vi prometto cose sorprendenti. Alla prossima!

Fabio Grimaldi

NOTA

ESCLUSIONI SECONDO GAUSS

Secondo calcoli non supportati da teoremi ma da dati sperimentali e dalla sensibilità fuori discussione di Gauss, per numeri che non eccedono il milione, sei o sette esclusioni sono sufficienti. Per numeri con otto o nove cifre nove o dieci esclusioni bastano.

Biblioteca

HACKER 5.0

Essere Hacker significa scoprire costantemente i propri limiti e superarli alla ricerca di sempre nuove ed emozionanti sfide. È una filosofia di vita prima che un argomento legato all'informartica. Il Libro Hacker 5.0 è un libro pratico che affronta le tecniche di sicurezza, eppure le sue pagine trasmettono un messaggio chiaro "Imparare a pensare come un Hacker", comprendere cosa vuol dire sicurezza, sapere come reagire in caso di difficoltà. Il libro si snoda in 670 pagine dense di contenuti che affrontano le tematiche dell'hacking da tutti i punti di vista. Si va dagli attacchi alle password alle insicurezze contenute nel codice HTML per affrontare infine argomenti molto recenti come il Voice Over IP e l'attacco alla reti Wireless. Si tratta di un libro davvero ben fatto, denso ed affascinante, scritto in modo tale da non appesantire la lettura con informazioni ridondanti ma andando invece direttamente al sodo, affrontando la sicurezza entrando direttamente nel cuore del problema.

Difficoltà: Bassa • **Autore:** Stuart McClure, Joel Scambray, George Kurtz • **Editore:** Apogeo • **ISBN:** 88-503-2362-X • **Anno di pubblicazione:** 2005 • **Lingua:** Italiana • **Pagine:** 670 • **Prezzo:** € 45,00

CSS GUIDA COMPLETA

I fogli di stile sono stati un'eccezionale novità al momento della loro uscita. Hanno radicalmente cambiato il modo di progettare il web, per poi nel tempo diventare uno strumento utile anche in altri tipi di applicazioni. Oggi si percepiscono alla base di ogni interfaccia grafica che si rispetti, e la loro evoluzione va di pari passo con quella effettuata dal web. IL libro di Gianluca Troiani è un riferimento certo, sicuro e approfondito sia per chi vuole imparare CSS ma anche per gli autori esperti che vogliono utilizzare le funzionalità avanzate dei fogli di stile, vogliono usarli per sviluppare funzionalità dinamiche, vogliono sfruttarne tutte le potenzialità. Si va dalle prime nozioni fino allo sviluppo di due progetti completi, passano per le tecniche più disparate, dall'uso dei menu all'uso dei filtri e delle condizioni alla gestione delle immagini. Infine si da uno sguardo a CSS3, futuro non troppo lontano dei fogli di stile.

Difficoltà: Media • **Autore:** Gianluca Troiani • **Editore:** Apogeo • **ISBN:** 88-503-2369-7 • **Anno di pubblicazione:** 2005 • **Lingua:** Italiana • **Pagine:** 480 • **Prezzo:** € 35,00

DATI E APPLICAZIONI PER IL WEB

Un libro molto particolare questo. Si pone infatti, come una guida alla modellazione di strutture dati per un settore dove nessuna regola o quasi nessuna è certa, quella del web. Eppure pensandoci con maggiore lucidità si intuisce facilmente come proprio il web sia un enorme concentratore di dati e come siano proprio le applicazioni web ad essere quelle che maggiormente sfruttano l'interfacciamento con database. Allora perchè stupirsi di un libro che spiega come modellare le proprie strutture in modo efficiente, per una più facile gestione e manutenzione delle applicazioni Web? Il libro, per quanto affronti un tema ostico è scritto in linguaggio facile e comprensibile e utilizza grafici e immagini che schematizzano in modo efficiente le tecniche di modellazione, ecco perché si pone come una guida interessante, per chiunque voglia affrontare il web in modo professionale.

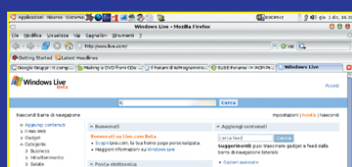
Difficoltà: Alta • **Autore:** Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai, Marisella Matera • **Editore:** McGrawHill • **ISBN:** 88-386-6138-3 • **Anno di pubblicazione:** 2003 • **Lingua:** italiana • **Pagine:** 526 • **Prezzo:** € 37,00

ON LINE

JAVASTAFF

Un sito relativamente nuovo, che negli ultimi tempi però ha effettuato un salto di qualità introducendo informazioni e articoli molto appetibili per i programmatori. Le tecnologie trattate sono sempre innovative ed il taglio molto comprensibile.
http://www.javastaff.com

GOOGLE BASE

Si tratta dell'ennesimo servizio innovativo offerto da Google. Questa volta è un frontend verso un database di proporzioni bibliche, all'interno del quale potete introdurre un qualunque tipo di contenuto e renderlo disponibile a tutti gli altri fruitori del servizio, che ovviamente possono ricercare le informazioni esattamente come in un grande database.
http://base.google.com

MICROSOFT LIVE

È un portale di Microsoft, ancora in Beta testing, che nelle intenzioni dovrebbe diventare un fornitore di email per chi ha già un nome a dominio. In sostanza chi ha un nome a dominio riceve 20 email configurabili con il proprio nome di domino e 200 mb di spazio. Una sorta di Gmail, ridimensionato ma associabile al proprio dominio anziché ad uno generico
http://www.live.com